

AD-A228 486

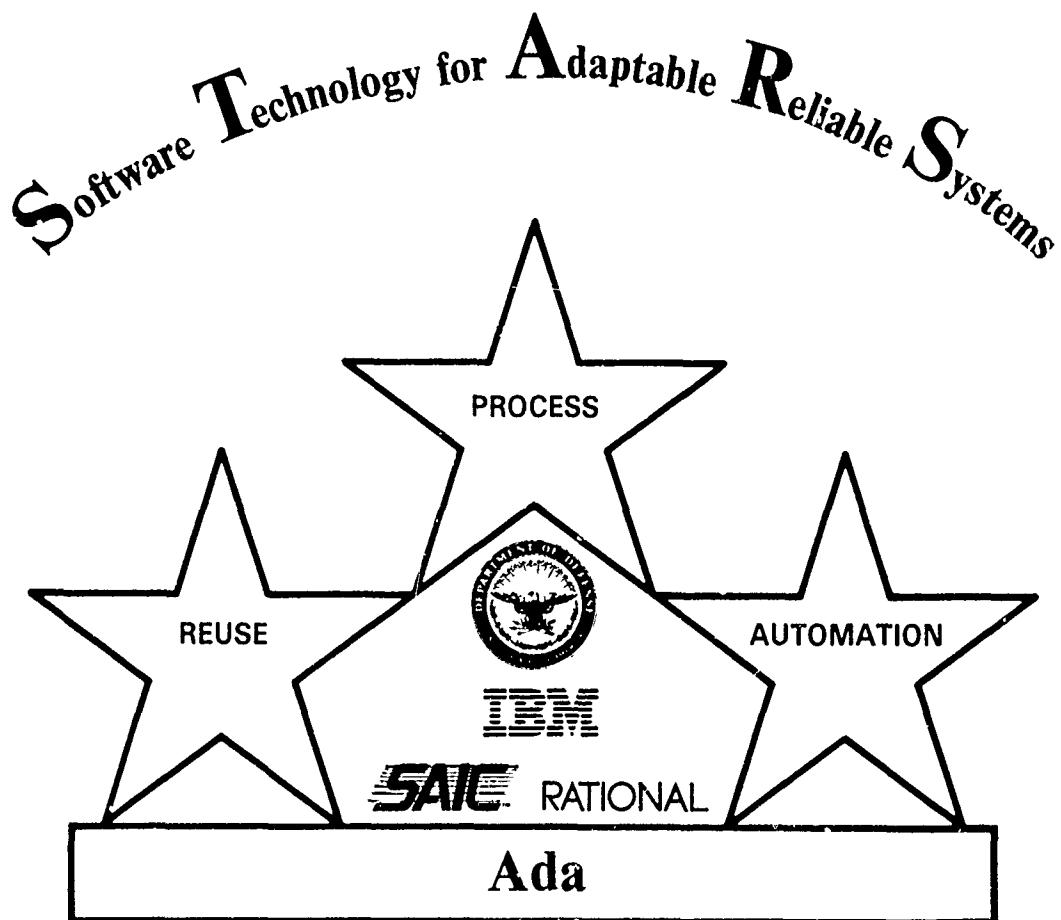
DTIC FILE COPY

Document No. 0080B-001  
17 March 1989

(2)

# STARS Technical Plan Analysis (Final)

for the



Contract No. F19628-88-D-0032

CDRL Sequence No. 0080B

17 March 1989

DTIC  
ELECTE  
S NOV 09 1990  
S C B D

DISTRIBUTION STATEMENT  
Approved for public release  
Distribution Unlimited

# REPORT DOCUMENTATION PAGE

Form Approved  
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project 0704-0188, Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED	
	March 17, 1989	Final	
4. TITLE AND SUBTITLE STARS Technical Plan Analysis (Final)			5. FUNDING NUMBERS C: F19628-88-D-0032
6. AUTHOR(S) R. Ekman			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) IBM Federal Sector Division 800 N. Frederick Avenue Gaithersburg, MD 20879		8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Electronic Systems Division Air Force Systems Command, USAF Hanscom AFB, MA 01731-5000		10. SPONSORING / MONITORING AGENCY REPORT NUMBER CDRL Sequence No. 0080B	
11. SUPPLEMENTARY NOTES			
12a. DISTRIBUTION / AVAILABILITY STATEMENT		12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) The revised top-level technical program plan for the STARS program. It describes the goals of the program, the technical approach to achieve these goals, the products to be delivered, and the product milestones and funding requirements.			
14. SUBJECT TERMS STARS			15. NUMBER OF PAGES 61
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

**STARS Technical Plan Analysis (Final)  
for the  
Software Technology for Adaptable, Reliable Systems  
(STARS) Program**

**Contract No. F19628-88-D-0032**

**CDRL Sequence No. 0080B**



**17 March 1989**

<b>Accession For</b>	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
<b>Justification</b>	
By <i>per letter</i>	
<b>Distribution/</b>	
<b>Availability Codes</b>	
<b>Dist</b>	<b>Avail and/or Special</b>
A-1	

Prepared for:

Electronic Systems Division  
Air Force Systems Command, USAF  
Hanscom AFB, MA 01731-5000

Prepared by:

IBM Systems Integration Division  
18100 Frederick Pike  
Gaithersburg, MD 20879

## Abstract

The current STARS Technical Program Plan (TPP) was published in August 1986. It contained a general plan to address the growing complexity and cost of mission critical software and specific guidance for STARS program participants. Since that time, new technologies have emerged and software development methods have evolved.

The STARS Prime Contractor Q1 task was established to address the shortcomings of the TPP. The primary objectives of the Q1 task were to analyze and republish the TPP. This report (IBM STARS CDRL Sequence Number 0080B) contains the revision of the TPP as an appendix. The body of the report contains sections on the research, analysis, and conclusions of the Q1 task.

The overall organization and content of the 1986 TPP remains appropriate and useful. Minor changes have been made to some sections and lists. It has also been updated to reflect recent developments, especially the results of the STARS foundation tasks, shadow programs, and the recently awarded prime contracts.

The 1986 TPP contained a broad approach to improving software development. It addressed the Ada language issues, development life cycle, standardization, and relationship to other programs. In the near term, the STARS program is focusing on the software development life cycle and removing the inhibitors to software reuse. This emphasis has been incorporated in the revised TPP, while retaining the comprehensive nature of the 1986 TPP.

The ideas in this report were collected through discussions with participants in the STARS program, literature research, and review of STARS repository documents. Material from recent presentations on the direction of the STARS program were also used. Referenced documents and supporting documents are listed in the bibliography.

# Table of Contents

<b>Introduction</b> .....	1
Scope .....	1
Task Description .....	1
<b>Research</b> .....	2
Major Influences on MCCR Software Development .....	2
Definition of Productivity .....	3
<b>Analysis</b> .....	5
STARS Program Documentation .....	5
General Changes .....	5
Specific Changes .....	6
Potential Changes .....	8
<b>Conclusions</b> .....	9
Lessons Learned .....	9
Recommendations .....	9
<b>Acronyms</b> .....	11
<b>References</b> .....	12
STARS Documents .....	12
Industry Publications .....	12
<b>Appendix A. Revised TPP</b> .....	14

# Introduction

## *Scope*

This report, IBM Contract Data Requirements List (CDRL) Sequence Number 0080B, is the final deliverable defined in the STARS Prime Contractor Delivery Order, Statement of Work, Paragraph 3.0, Task Q1: Technical Plan Analysis [RFP87].

In this report, IBM presents an analysis of the STARS Technical Program Plan (TPP), 6 August 1986 [TPP86], and the conclusions of the STARS Q1 task. As an appendix to this report, IBM presents a complete revision of the TPP. This revision is based on the research, analysis, and review conducted during the Q1 task performance period. Preliminary analysis and revision comments were presented in the informal report IBM CDRL 0080A, 17 December 1988 [IBM0080A].

Initially, the Q1 task had only one CDRL (Sequence Number 0080). During the task performance period, it was determined that a better solution to the task problem was to have the period of performance extended and an additional CDRL defined. The original CDRL 0080 was changed to CDRL 0080A and the new CDRL was designated CDRL 0080B. This change was agreed to by the STARS Joint Program Office in November 1988.

## *Task Description*

The STARS TPP is the highest level technical document in the suite of STARS program documentation and deliverables. It was last published in 1986. Since that time, the following developments have signalled that a review and update of the TPP is warranted:

- Completion of STARS Foundation Tasks.
- Completion of STARS Shadow Programs.
- Awarding of STARS Prime Tasks.
- Deliveries of initial STARS Prime documents.
- Changes in software development platforms and environments.
- Development and refinement of software development technologies.
- Continued pressures on DoD research funding.

To update the plan, the STARS prime Q1 task was established. This task was divided into two subtasks:

- Analyze the TPP and research related technical issues, and then
- Revise and republish the TPP.

Both subtasks required a delivered report on the subtask research and analysis. The first subtask report included an annotated copy of the current TPP. The second report, this report, includes a final revision of the TPP.

## Research

The following research was used to augment the analysis of the TPP and support judgments made during the revision.

### *Major Influences on MCCR Software Development*

Significant changes are taking place to the tools and technologies that will be used by software engineers in the 1990s. Many of these changes will become reasonably mature during the STARS program. As stated in the TPP, the STARS program must use and potentially influence these changes to address the needs of Mission Critical Computer Resources (MCCR).

The following technology areas have been identified through Q1 task research as having the greatest influence on software engineering in the 1990s.

- High Powered Workstations - Almost all software development in the 1990s will be accomplished from workstation based systems. The best workstations will be high powered system (by today's standard) with a large high resolution screen, extensive keyboard, infinite virtual memory, gigabytes of local storage, and auxiliary input devices such as a mouse and a scanner. The targets for the software development activity will be based on the needs of the application (i.e., embedded micros, unique chips, commercial mainframes). The program requirements to develop on target machines will be diminished.
- High Quality Documentation - Engineers currently spend at least 60% of their effort in non-code activities, such as documentation and general communication [COST]. Mono-font impact printer documentation will disappear as electrostatic printers become common and cost effective. Screen and printer interfaces will merge and WYSIWYG printing will be common. The ease of use and increased processing speed will reduce the documentation burden on the engineers. Image storage, retrieval, and manipulation will begin to enter software development environments but will not have a major effect.
- Extensive Networking - Government and industry networks have already significantly altered the software engineer's approach to software development. Electronic mail, office automation software, bulletin boards, electronic conferences, repositories, dial access, and local area networks will continue to provide an ever expanding universe of information. The availability of these systems will have a major influence on human communications and DoD program operations. Electronic delivery, electronic peer review, and shared technology will become common and move the industry toward a paper-less environment [PAPER].
- Graphical Software Design - This area will mature in the next few years [DESIGN]. The major breakthrough will come when software can be maintained in a form that permits multiple views, such as graphic and code, and when changes in any view will be reflected in the other views automatically.
- New Data Organizations - Improved data organizations are required to support the vast amounts of information underlying the other changing technologies, such as configuration management and documentation. Entity-Relationship-Attribute (ERA) data systems are one type of data organization that can meet the need. Object oriented development is a natural fit for ERA systems. Hypertext technology and related products fall in this same category of improved ways of looking at information [HYPER].

- Standard Interfaces - The layering of the software development environment and the standardization of interfaces between the layers will allow for the creation of plug compatible software parts. Flexible, extensible, and portable architectures will develop when interfaces and layers are better established. These architectures will support customization to different application domains [ENV].
- Development Life Cycle Changes - Software-First life cycle, rapid prototyping, Computer Aided Acquisition and Logistics Support (CALS), and related developments [SPIRAL] will have significant direct effect on the productivity of software engineers. New procurement processes must be used to allow the new technologies to flourish [PROD]. Legal issues, such as data rights and liability, are potential roadblocks and must be resolved in parallel with the technology issues [LEGAL].
- Human Behaviors - The behaviors of the participants in the system development process are the most powerful potential for productivity improvements [BEHAVE]. Engineers, procurement officers, users, customers, contractors, and vendors working together with an enhanced problem domain understanding will make significant reductions in system costs and time to deployment.

## *Definition of Productivity*

Since the major objective of the STARS program is to increase software engineering productivity, a research activity was conducted under the Q1 task to refine this objective. Engineers in IBM and the industry were asked to give their opinions on software productivity. In addition, industry and government publications were reviewed for definitions and experiences [COST]. The following is a brief discussion resulting from this research.

The current definition of software productivity in IBM Systems Integration Division is product developed per labor month of effort expended, normalized to a standard work scope. The product developed is measured in units of source lines of code (SLOC). The labor expended is measured in labor months (LM).

This definition requires a further definition of SLOC, scope of work, and the life cycle model. For example, SLOC could be defined as lines of code or language statements. Scope of work could be defined as total life of project or just software development. In some cases, lines of documentation or design are also quoted (SLOD). These issues will not be analyzed here, since most projects make these refinements as required, although they are not consistent across projects.

The basic algorithm for software productivity is given as:

$$\text{SLOC/LM}$$

The increasing use of existing code (reuse) has created a modification to the definition of SLOC. Total SLOC is characterized as "new"(NSLOC), "reused"(RSLOC), or "reused but changed"(CSLOC). Value weighting factors (WR and WC) are used for each class other than new. Further, complexity coefficients (CCN, CCR, and CCC) are also used. So total SLOC is then defined as equivalent SLOC:

$$\text{ESLOC} = (\text{NSLOC} * \text{CCN}) + (\text{RSLOC} * \text{WR} * \text{CCR}) + (\text{CSLOC} * \text{WC} * \text{CCC})$$

Another idea frequently mentioned is that productivity should be measured only in the delivered product (DSLOC). The previous ESLOC definition assumed that SLOC was the total code written, regardless of whether it was delivered.

There are other measures of productivity that have merit and could be used:

- Function Points (FP/LM) - This method defines productivity based on the functions addressed by the software development activity. A related method is based on feature points.
- Delivered Machine Instructions (DMI/LM) - Since ultimately all executable software ends up as machine instructions, these machine instructions could be used to measure productivity.

This method requires an additional factor to account for data within the machine instruction unit.

- From a pragmatic point of view, productivity could be defined as the cost to produce the product (cost/time). This is frequently referred to as "burn rate" and proceduralized in the "earned value" process. This method also has the advantage of accounting for activities not directly associated with coding, such as documentation and meetings.

The research resulted in the definition that productivity is the rate of producing software products, but the measured value over time could be against many different forms of the product [PROG].

## **Analysis**

In 1986, the STARS program documentation was completely revised and rebaselined. Since then, this documentation has been augmented by the deliverables of the various STARS program contracts.

The first task in the analysis of the TPP was to decide where in the structure of documents the TPP belonged and what was the appropriate content. The TPP was then reviewed for general content and organization. This task identified which sections should be deleted, moved to other documents, moved to other areas of the TPP, and finally, which sections should be augmented. The final task was to make minor additions and modifications.

## ***STARS Program Documentation***

The highest level document in the STARS hierarchy of documentation is the STARS Program Management Plan (PMP). The PMP, based on DoD directives and authorizations, contains programmatic, financial, and organizational information [PMP]. It also provides direction for industry contracting and service relationships.

The TPP, subordinate to the PMP, provides overall technical direction for the STARS program. It describes functional objectives that are useful in describing the program to the DoD and industry. It contains the description of the program elements: prime contracts, foundation tasks, and shadow programs.

Each program element has developed its own suite of documents. The foundation and shadow programs are now effectively completed. Their work, while important for the success of the STARS program, has little influence on the technical content revision of the TPP.

The prime contractors have developed a consolidated plan that addresses the technical direction of the continuing STARS program [IBM0070]. They have also developed documents that describe specific technology subjects in great detail [IBM0110, IBM0360]. In addition, the prime contractors have developed innovative ideas that relate to the TPP through presentations, peer review, and meetings.

## ***General Changes***

The following general changes have been incorporated in the revised TPP:

- Material that was replicated in the PMP has been removed.
- Specifics on future delivery dates and content have been removed.
- Program operations and financial material has been removed.
- Updates were made to reflect the completion of STARS shadow and foundation tasks and the selection of prime contractors.
- The importance of products and environments has been de-emphasized. This position is substantiated by numerous articles on software engineering [BULLET, COST, PROD]. The

software "crisis" will be relieved by changes in human behavior and not hardware or software environments [BEHAVE].

- Life cycle support and standard interfaces have been emphasized. This is documented in the STARS Consolidated Plans [IBM0070] and referenced articles on software environments [ENV]. The environment must have capabilities as defined in the Q3 task summary [IBM0110].

## *Specific Changes*

The following specific changes have been incorporated into the revised TPP:

- Section 3.2 Productivity Improving Opportunities - Specific productivity goals and measurements were removed. Emphasis on a particular form of measured productivity or specific improvement factor will dilute the efforts required to meet the objectives of the program. Because of the vagueness of productivity, the TPP should not quote a specific figure for improvement, but rather assume it will be easily recognized.
- Section 3.2.3 Software Reuse - A chart on the inhibitors to software reuse was added.
- Section 3.3 Software-First Processes The software-first process definition was given more detail. A figure on the Software-First Life Cycle was added.
- Section 3.4 Standardization Support - The prime contractor's consolidated plan generic architecture was used to replace the figure on generic interfaces.
- Section 4.2 Shadow Demonstrations - The criteria for the shadow programs selection were made into a list.
- Section 4.3.2 MCCR Software Engineering Environments - The list of framework principles was reorganized. Items were divided among the three sections: Frameworks, Process Control, and Database.
- Section 4.4.1.2 Standards - Items were added to the list: SQL, OSI, POSIX, IRDS, X-Windows.
- Section 4.5 Breakthrough Initiatives - The concepts in risk reduction have been focused on breakthrough initiatives. A figure has been added.
- Section 5 Technical Guidance - The use of an Ada command language has been soften. Guidance on the following items has been added:
  - documentation
  - networks/repositories
  - data rights
  - peer review
- Appendices - Two appendices were added to hold the details of the Shadow Projects and the Prime Contracts.

The following chart maps the 1986 TPP headers to the revised TPP headers. The "Revision Bar" indicates the sections that received substantial modifications or additions.

1986 TPP Headers	Revision Bar	Revised TPP Headers
EXECUTIVE SUMMARY		EXECUTIVE SUMMARY
SECTION 1 - BACKGROUND		SECTION 1 - BACKGROUND
1.1 Software Engineering Criticality		1.1 Software Engineering Criticality
1.2 Scope of the STARS Program		1.2 Scope of the STARS Program
1.3 Stars Funding		1.3 STARS Program Funding
SECTION 2 - OBJECTIVE		SECTION 2 - OBJECTIVE
2.1 Defense Systems of the Nineties		2.1 Defense Systems of the Nineties
2.2 Capabilities Needed		2.2 Capabilities Needed
SECTION 3 - TECHNICAL STRATEGY		SECTION 3 - TECHNICAL STRATEGY
3.1 Leverage Approach		3.1 Leverage Approach
3.2 Productivity-improving Opportunities		3.2 Productivity Improving Opportunities
3.3 Software-first Processes		3.3 Software-First Processes
3.4 Ada - the Language of Choice		(moved to 3.2.1)
3.5 Standardization Support		3.4 Standardization Support
3.6 Relationships to Certain Other Programs		3.5 Relationships to Other Programs
3.7 Restructuring of the FY 1986 Program		3.6 Changes to the STARS Program
SECTION 4 - TECHNICAL PROGRAM		SECTION 4 - TECHNICAL PROGRAM
4.1 Program Structure		4.1 Program Structure
4.2 Shadow Demonstrations		4.2 Shadow Demonstrations
4.2.1 1985-Planned Applications		(moved to Appendix B)
4.2.2 FY 1986 Shadow Projects		(moved to Appendix B)
4.2.3 Shadow Criteria		(moved to 4.2)
4.3 Product Development		4.3 Product Development
4.3.1 Technology Integration		4.3.1 Technology Integration
4.3.2 MCCR Software Engineering Environments		4.3.2 MCCR Software Engineering Environments
4.3.2.1 Frameworks		4.3.2.1 Frameworks
4.3.2.2 Control Processes		4.3.2.2 Process Control
4.3.2.3 Database		4.3.2.3 Database
4.4 Technology Development		4.4 Technology Development
4.4.1 Adaptable Software Technology		4.4.1 Adaptable Software Technology
4.4.1.1 Foundation Ada Capabilities		4.4.1.1 Foundation Ada Capabilities
4.4.1.2 Standards		4.4.1.2 Standards
4.4.1.3 Repository		4.4.1.3 Repository
4.4.1.4 Design by Successive Refinement		4.4.1.4 Design by Successive Refinement
4.4.2 Reliable Software Technology		4.4.2 Reliable Software Technology
4.4.2.1 Computer Aided Methods		4.4.2.1 Computer Aided Methods
4.4.2.2 Domain Integration		4.4.2.2 Domain Integration
4.4.2.3 Formal Methods		4.4.2.3 Formal Methods
4.5 Risk Reduction		4.5 Breakthrough Initiatives
SECTION 5 - TECHNICAL GUIDANCE		SECTION 5 - TECHNICAL GUIDANCE
SECTION 6 - PRODUCTS AND MILESTONES		SECTION 6 - PRODUCTS AND MILESTONES
6.1 Shadow MCCR Systems		6.1 Shadow MCCR Systems
6.2 Software Development Environments		6.2 Software Development Environments
6.3 Technology Development		6.3 Technology Development
6.4 Fundamental Research		6.4 Fundamental Research
6.5 Detailed Milestones		(moved to 6.2)
SECTION 7 - REFERENCES		SECTION 7 - REFERENCES
APPENDIX A - FOUNDATION CAPABILITIES		APPENDIX A - FOUNDATION CAPABILITIES
A.1 General Fragments		A.1 General Fragments
A.1.1 Command Languages		A.1.1 Command Languages
A.1.2 Document/Text Preparation		A.1.2 Document/Text Preparation
A.1.3 Database Support		A.1.3 Database Support
A.1.4 Operating System Fragments		A.1.4 Operating System Fragments
A.1.5 Graphics Support		A.1.5 Graphics Support
A.1.6 Network Support		A.1.6 Network Support
A.2 Automated Environment Fragments		A.2 Automated Environment Fragments
A.2.1 Design Description and Analysis		A.2.1 Design Description and Analysis
		A.3 Foundation Deliveries
		APPENDIX B - SHADOW PROJECTS
		APPENDIX C - PRIME CONTRACTORS
		C.1 Consolidated Plan
		C.2 First Increment Deliveries

## *Potential Changes*

The following changes were identified as desirable in the preliminary analysis, but have not been incorporated into the TPP at this time:

- A more detailed commercial motivation plan could be added to Leverage Approach section (3.1). The acquisition process and legal issues could be further explained. Delaying this detail was warranted, since the government is currently engaged in major revisions to these policies.
- A clearer definition of the environment framework could be added to the Environment section (4.3.2). This definition will be developed by the prime contractors and could be added in a future revision.
- More detail on the results of the program elements (foundations, shadows, primes) could be added to the Appendices. This material was not generally available and collection of it was lower priority than the revising the overall document.
- Justification for the technical guidance section should be established. Industry experiments and articles supporting the guidance could be referenced. This was also a low priority task and may benefit from the current prime contractor efforts.

## **Conclusions**

Generally, the TPP was acceptable and required only a revision to bring it up-to-date. The plan was balanced, reasonable, understandable, and not overly ambitious. The section headings and order of topics only required minor modifications. Major additions were made based on the STARS prime contractor efforts. The updated TPP is included in this report as an appendix.

## ***Lessons Learned***

Besides updating the TPP, the Q1 task activities have resulted in an increased appreciation of the following issues:

- Improved communications, on the human and hardware level, is key to solving the difficult problems facing the software engineer. The STARS prime contractors' repositories and networks are now established. These systems, along with corporate and DoD networks, will provide access to STARS developed data to a significant number of engineers. This form of networking will be a major aspect of reducing elapsed time in system development.
- Face-to-face communications have been very successful in making progress on difficult issues. The work of the STARS prime contractors on the Consolidated Plan [IBM0070] and the Consolidated Reusability Guidelines [IBM0380] are examples of how we can make progress. The STARS workshops have also been successful in developing a STARS community and transferring experience among the participants.
- Making suggestions for updating a document and actually updating the document are two different activities, much the same as code reviews and code modifications. It was easy to comment on the areas to change in the TPP. It has been difficult to establish consensus on the changes, collect detail for the changes, and then publish the revision.

## ***Recommendations***

The following are specific recommendations resulting from the Q1 task activities:

- The TPP, along with other program wide plans, should be reviewed and republished annually. This process should be coordinated through the STARS Joint Program Office.
- The Q1 task updated only the TPP, but to be intellectually complete, the PMP should also be revised. Given the work started for the TPP, this additional task should not be as large as the Q1 task, but would require effort from the STARS Joint Program Office and contacts with the various DoD project agencies and funding organizations.
- The following considerations have helped in revising the TPP, and should be addressed for future revisions:
  - The TPP, PMP, consolidated plans, direction memoranda, and related program wide documents should be kept on-line and in a form for future electronic processing.
  - A historical trail of program changes and results should be maintained and optimally kept on-line for easier collection and summarization of program status. This information should include participant names, financial expenditures, and miscellaneous notes.

- The STARS community members must participate in the revision process through electronic on-line methods.
- As identified in the research task, many new technologies will be available for the software engineer in the 1990s. The STARS program should take advantage of these developments, but not become overly involved in them since they will evolve with existing industry efforts. The STARS program should concentrate in technology thrusts that are not covered by industry interests but that are critical for DoD programs.
- As a community, we need to increase our use of improved methods to deal with the technology explosion. The number of new products, articles, and ideas that are pouring out of the computer industry is staggering. It has become impossible for the engineers to digest all this material. Enhanced electronic media centers and expanded on-line research database systems will satisfy some of this need. Legal and business inhibitors to these on-line capabilities should be addressed and reduced.

## Acronyms

<i>Acronym</i>	<i>Meaning</i>
APSE	Ada Programming Support Environment
CALS	Computer-Aided Acquisition and Logistics Support
CDRL	Contract Data Requirements List
DoD	(United States) Department of Defense
DARPA	Defense Advanced Research Projects Agency
DIANA	Descriptive Intermediate Attributed Notation for Ada
ERA	Entity-Relationship-Attribute
IBM	International Business Machines
IRDS	Information Resource Dictionary System
LM	Labor Month
MCCR	Mission Critical Computer Resource
OSI	Open Systems Interconnect
PDL	Program Design Language
PMP	(STARS) Program Management Plan
POSIX	Portable Operating Systems
SGML	Standard Generalized Markup Language
SLOC	Source Lines of Code
SQL	Structured Query Language
STARS	Software Technology for Adaptable, Reliable Systems
TPP	(STARS) Technical Program Plan
WYSIWYG	What-you-see-is-what-you-get

## References

### *STARS Documents*

[TPP86] United States Department of Defense, STARS Joint Program Office, *STARS Technical Program Plan*, August 6, 1986.

[PMP86] United States Department of Defense, STARS Joint Program Office, *STARS Program Management Plan*, August 6, 1986.

[IBM0070] IBM Systems Integration Division, *Consolidated Technical Development Plan for the Software Technology for Adaptable, Reliable Systems (STARS) Competing Prime Contractors*, CDRL Sequence No. 0070, November 11, 1988.

[IBM0080A] IBM Systems Integration Division, *STARS Technical Plan Analysis*, CDRL Sequence No. 0080A, December 17, 1988.

[IBM0110] IBM Systems Integration Division, *Environment Capability Matrix*, CDRL Sequence No. 0110, March 17, 1989.

[IBM0360] IBM Systems Integration Division, *Reusability Guidelines*, CDRL Sequence No. 0360, December 17, 1988.

[RFP87] United States Department of Defense, Department of the Air Force, *STARS Competing Primes Lead Contracts Request For Proposal*, F19628-88-R-0011, November 5, 1987.

### *Industry Publications*

[SPIRAL] Boehm, B.W., "A Spiral Model of Software Development and Maintenance", *IEEE Transactions on Software Engineering*, Vol. 14, No. 10, October 1988.

[COST] Boehm, B.W. and Papaccio, P.N., "Understanding and Controlling Software Costs", *IEEE Computer*, Vol. 21, No. 5, May 1988.

[BULLET] Brooks, F.P., "No Silver Bullet, Essence and Accidents of Software Engineering", *IEEE Computer*, Vol. 20, No. 4, April 1987.

[BEHAVE] Curtis, B., Krasner, H., and Iscoe, N., "A Field Study of the Software Design Process for Large Systems", *ACM Communications*, Vol. 31, No. 11, November 1988.

[DESIGN] Davis, A., "A Comparison of Techniques for the Specification of External System Behavior", *ACM Communications*, Vol. 31, No. 9, September 1988.

[HYPER] Fiderio, J. "A Grand Vision", *BYTE*, Vol. 13, No. 10, October 1988.

[PAPER] Hansen, W.J. and Hass, C., "Reading and Writing with Computers: A Framework for Explaining Differences in Performance", *ACM Communications*, Vol. 31, No. 9, September 1988.

[PROG] Jones, C., *Programming Productivity*, McGraw-Hill Book Company, 1986.

[ENV] Penedo, Maria H., and William E. Riddle, "Software Engineering Environment Architectures, Guest Editors' Introduction", *IEEE Transactions on Software Engineering*, Vol. 14, No. 6, June 1988.

[PROD] Port, O., "How the New Math of Productivity Adds Up", *Business Week*, June 6, 1988.

[LEGAL] Shore, J., "Why I Never Met a Programmer I Could Trust", *ACM Communications*, Vol. 31, No. 4, April 1988.

## **Appendix A. Revised TPP**

The following document is a revised publication of the STARS Technical Program Plan (TPP). Reviewers of the revised document are encouraged to make annotations and forward them to:

Robert W. Ekman  
IBM Systems Integration Division  
18100 Frederick Pike  
Gaithersburg, MD 20879  
phone: (301) 240-6431  
IBM VNET: rckvml(ekmanb)  
ARPANet: ekmanr@ajpo.sei.cmu.edu  
IBM/SAIC Repository: stars::ekmanb

# **SOFTWARE TECHNOLOGY FOR ADAPTABLE, RELIABLE SYSTEMS (STARS) TECHNICAL PROGRAM PLAN**

*Document Number STARS TPP*

*17 MARCH 1989*

STARS Joint Program Office  
Defense Advanced Research Projects Agency (DARPA)  
Rosslyn, VA 22209

# Preface

Software Technology for Adaptable, Reliable Systems, or STARS, is the Defense Department's program to achieve dramatic improvements in software quality and to mitigate runaway software costs. This technical plan was originally prepared by the STARS program director on 6 August 1986, and approved by the STARS Executive Committee. The plan was revised 17 March 1989 with program updates and technical refinement. An [R] next to the section header is used to denote an added or substantially revised section.

The STARS Charter [1] signed by the Under Secretary of Defense for Research and Engineering (USDRE) on 1 November 1984, the Deputy Secretary of Defense program clarification memorandum [2], signed on 12 August 1985, and the requirements document prepared by the Services [3] provide the approved high level statements of STARS program requirements, objectives and approach. The technical guidance from references 1 and 2 is presented next in outline form to summarize the approved basis for this STARS technical plan.

- REQUIREMENTS

- Future weapon systems require:
  - ▲ Adaptability and reliability.
  - ▲ New and enhanced capabilities.
  - ▲ Significant quantities of software.
- Software cost projections require:
  - ▲ Unusual efforts to mitigate rising costs.

- OBJECTIVE

- Long term savings and reliability benefits.
- Major advances in the software engineering process.
- Dramatic improvements in:
  - ▲ Software engineering productivity.
  - ▲ Software quality and reliability.
  - ▲ Time and cost of Defense software.
- Transition of new processes into use.

- APPROACH

- Apply sufficient resources in a concentrated R&D effort.
- Exploit new software and systems opportunities for:
  - ▲ Commonality.
  - ▲ Standardization.
  - ▲ Portability.
  - ▲ Reusability.
- Use a focused management approach.
- Assign Service and Defense Agency organizations responsibility for technical direction and contract management.
- Develop major products under competitive industry contracts.

This technical plan unifies and focuses the STARS program by emphasizing the use of Ada and related software engineering technology, introducing the software-first systems acquisition approach, and establishing a strong industry leadership role in STARS technology development.

Specifically, STARS will:

- Develop and demonstrate a software-first technology with special emphasis on adaptability to reduce software costs and reliability to improve software quality. The current "waterfall" method is not sufficient to produce the productivity improvements that the STARS program has as an objective. This development of a new software life cycle will have far reaching implication in the development and acquisition of Mission Critical Computer Resources (MCCR) systems.
- Develop and support an adaptable software technology through several initiatives:
  - Develop a significant foundation of reusable Ada software for MCCR mission applications and software engineering support.
  - Unify a set of commercial functional interface standards in Ada as a basis for software component compatibility, adaptability and reuse.
  - Provide a MILNET-accessible repository with appropriate access controls to support software reuse. Because STARS will seek to demonstrate and support software reusability opportunities to reduce mission application software costs, the repository will include a significant quantity of mission application software that can be used to evaluate and advance software development approaches for reusable software.
- Develop and demonstrate a reliable software technology through several initiatives:
  - Develop and demonstrate computer aided tools to automate and unify software system definition, specification, design, coding, testing, maintenance and documentation, and to provide software life cycle configuration control over all aspects of the software process. Validation capabilities will be included.
  - Demonstrate the feasibility of extending the mechanisms used by Ada compiler builders in linking Ada package specifications and package bodies to provide comparable processes for assuring completeness and consistency between other phases of the life cycle process.
  - Contribute to the development of formal verification methods and supporting tools for MCCR systems developed in Ada.
- Pioneer and demonstrate a new way to treat high-risk software issues in MCCR system acquisition.
- Deliver several prototype automated software engineering environments that ultimately become fully operational environments that run as distributed applications over networked heterogeneous hardware, exploit standard virtual interfaces, provide an object-oriented view to developers, provide configuration control of developed software, and support a design-by-successive-refinement software-first strategy.
- Demonstrate emerging STARS technology by sponsoring shadow projects with Service Systems Program Offices Shadow projects are implementations of operational mission software using emerging STARS technology. These projects will be conducted in parallel on a non-interference basis with active system development efforts by selected System Program Offices of the Services. Shadow projects will provide a pragmatic assessment of STARS progress and will help transition STARS technology into MCCR programs.

Written line-in, line-out comments with rationale are invited and should be forwarded to the Director of the STARS Joint Program Office, Defense Advanced Research Projects Agency, DARPA/DMO, 1400 Wilson Blvd., Rosslyn, VA 22209.

**Submitted:**

**Joseph S. Greene, Jr.**  
**Colonel, USAF**  
**Director, STARS**

# Table of Contents

<b>SECTION 1 - BACKGROUND</b>	1
1.1 Software Engineering Criticality	1
1.2 Scope of the STARS Program	1
1.3 Stars Funding [R]	2
<b>SECTION 2 - OBJECTIVE</b>	3
2.1 Defense Systems of the Nineties	3
2.2 Capabilities Needed	3
<b>SECTION 3 - TECHNICAL STRATEGY</b>	5
3.1 Leverage Approach [R]	5
3.2 Productivity Improving Opportunities [R]	5
3.2.1 Ada - The Language of Choice [R]	6
3.2.2 Labor Saving Tools [R]	6
3.2.3 Software Reuse [R]	7
3.3 Software-First Processes [R]	8
3.4 Standardization Support	10
3.5 Relationships to Other Programs	11
3.6 Changes to the STARS Program [R]	11
<b>SECTION 4 - TECHNICAL PROGRAM</b>	13
4.1 Program Structure	13
4.2 Shadow Demonstrations [R]	13
4.3 Product Development	14
4.3.1 Technology Integration	15
4.3.2 MCCR Software Engineering Environments	15
4.3.2.1 Frameworks [R]	15
4.3.2.2 Software Process Control [R]	16
4.3.2.3 Database [R]	16
4.4 Technology Development	16
4.4.1 Adaptable Software Technology	17
4.4.1.1 Foundation Ada Capabilities	17
4.4.1.2 Standards	18
4.4.1.3 Repository	19
4.4.1.4 Design by Successive Refinement	20
4.4.2 Reliable Software Technology	20
4.4.2.1 Computer Aided Methods	20
4.4.2.2 Domain Integration	20
4.4.2.3 Formal Methods	21
4.5 Breakthrough Initiatives [R]	21
<b>SECTION 5 - TECHNICAL GUIDANCE [R]</b>	23
<b>SECTION 6 - PRODUCTS AND MILESTONES</b>	25
6.1 Shadow MCCR Systems [R]	25
6.2 Software Development Environments [R]	25
6.3 Technology Development [R]	25
6.4 Fundamental Research [R]	26

<b>SECTION 7 - REFERENCES .....</b>	<b>27</b>
<b>SECTION 8 - ACRONYMS [R] .....</b>	<b>29</b>
<b>Appendix A. FOUNDATION PROGRAM .....</b>	<b>31</b>
<b>A.1 General Fragment Capabilities .....</b>	<b>31</b>
A.1.1 Command Languages .....	31
A.1.2 Document/Text Preparation .....	32
A.1.3 Database Support .....	32
A.1.4 Operating System Fragments .....	33
A.1.5 Graphics Support .....	33
A.1.6 Network Support .....	34
<b>A.2 Automated Environment Fragment Capabilities .....</b>	<b>34</b>
A.2.1 Design Description and Analysis .....	34
<b>A.3 Foundation Deliveries [R] .....</b>	<b>34</b>
<b>Appendix B. SHADOW PROJECTS [R] .....</b>	<b>36</b>
<b>Appendix C. PRIME CONTRACTORS [R] .....</b>	<b>37</b>
<b>B.1 Consolidated Plan [R] .....</b>	<b>37</b>
<b>B.2 First Increment Deliveries [R] .....</b>	<b>37</b>

# SECTION 1 - BACKGROUND

This document is the top-level technical program plan for the STARS program. It describes the objectives of the program, the technical approach to achieve the objectives, the products to be delivered, and the product milestones and funding requirements. The requirements and needs which drive the program may be found in the STARS program requirements document [3]. Program management is described in a separate management plan [4].

## *1.1 Software Engineering Criticality*

DoD's weapon and warfare support systems depend on Mission Critical Computer Resources (MCCR) not just for their initial operational capability, but also to provide the capability to respond to changing operational threats and employment doctrine. Future national security will require new and enhanced high-technology weapons, command control and intelligence gathering systems that will depend on significant quantities of software. Without fundamental change to the software engineering process, DoD's annual expenditures for MCCR are projected to increase from approximately \$3 billion in 1980 to \$11 billion in 1985 and then up to over \$30 billion in 1990 [5]. Only a 20 percent increase in software productivity by 1990 would equate to cost saving sufficient to buy a new ship for the Navy, 15 new fighter aircraft for the Air Force and a tank battalion for the Army every year. We should be able to do considerably better than 20 percent just by using Ada software engineering and institutionalizing software reuse. Today, however, software development and maintenance ability is the critical-path activity limiting progress in many DoD MCCR programs, from affordability, risk management and capability points of view.

Not only will future systems require more software at a higher aggregate cost, but the software will be responsible for providing more and more of the functionality in weapon systems. Much of the growth in the power and sophistication of U.S. weapon and warfare support systems has been due to the extensive application of computers, particularly hardware technology. Software productivity increases have simply been inadequate to meet the growing complexity and size of military systems. As a result, software problems continue to grow more serious. Symptoms of these problems are seen in slippages in weapon system development schedules, operational system failures, weapon system inability to meet changing requirements, and soaring life cycle costs [3, 6, and 7]. The Software Technology for Adaptable, Reliable Systems program, called STARS, is the DoD's program to achieve dramatic improvements in software quality and to mitigate runaway software costs.

## *1.2 Scope of the STARS Program*

With planning initiated in Calendar Year (CY) 1981 and continuing through Fiscal Year (FY) 1983, the STARS program experienced a slow and ambiguous start. In FY 1986, following major reviews [8, 9] STARS was redirected (See Section 3.6) and challenged to find and demonstrate ways to significantly reduce MCCR system software costs by CY 1991. New technical and management approaches were formulated to achieve major productivity enhancement. The new STARS technical approach will focus on use and extension of the Ada software engineering opportunities. The STARS management approach will exploit industry leadership in software research. Laboratories of the Military Departments will have a stronger role. STARS work will emphasize full and open competition continuing through the life of the program. STARS will foster the needed technology by funding research and development, integrating the contributions of many industry organizations and making those results conveniently available to the U.S. software industry. A research thrust

is required because the ability to produce such a software engineering approach does not now exist. However, DoD software initiatives over the past fifteen years support the feasibility of the planned software-first thrust with high expectation of significant near-term results.

STARS cannot solve the DoD's software problems by itself. The STARS program must be built upon active DoD programs and must leverage the commercial and industrial base. Tight coupling exists between STARS and technology-based programs within the DoD and other Government Agencies. These include the introduction and maintenance of the Ada computer programming language by the Ada Joint Program Office, the efforts of the Joint Logistics Commanders (JLCs) to improve software acquisition practices, the establishment of the Software Engineering Institute (SEI), the establishment of life cycle software engineering support centers by the Services, and the strategic computing initiatives by the Defense Advanced Research Projects Agency (DARPA).

### *1.3 Stars Funding [R]*

During the maturing process of the STARS program, the technical plans have been defined incrementally, with multiple objectives and directions. These plans have permitted the funding to be adjusted to meet the current plan. The current funding profile for the STARS program is defined in the STARS Program Management Plan [4].

## SECTION 2 - OBJECTIVE

The objective of the STARS program is to achieve a dramatic improvement in our ability to provide and support software meeting mission critical defense requirements. This improvement will be reflected in the cost, schedule, and quality of MCCR software. The program will seek major improvements by the early 1990s.

### *2.1 Defense Systems of the Nineties*

The STARS Technical Program Plan and implementation strategy have been developed based on a vision of the computing requirements for defense systems in the 1990's [6, 7]. By 1990, new mission-critical defense systems will exhibit the following characteristics:

- MCCR embedded in weapon systems will make routine use of multi-processors and networked or parallel architectures. Similarly, Command and Control systems will be based almost exclusively on distributed processing architectures.
- Data bases used by the commander will be distributed. Extensive use of computer networks rather than hierarchical structures will be used.
- The processing power of the computers used in defense systems will be increased by at least an order of magnitude. If the size and complexity of software grow at the same rate, the point will be reached beyond which continued implementation of software systems by past methods may become impractical.
- The amount of on-line computer-managed data will vastly increase. The automated systems supporting the commander must often sort through and reduce myriads of data to locate and present only the critical elements of information.
- The costs of the computer hardware used for software development and used in system acquisition and deployment will continue to decrease. The cost to develop and maintain software will be a principal factor affecting MCCR weapon system affordability and capability.

The STARS strategy contained in this plan responds to these critical factors affecting weapons system technology. The maturation of software technologies permitting the use of distributed computer networks, real time processing with multiprocessors, information protection for high integrity and trusted systems, and cost effective software generation concepts are topics of high interest.

### *2.2 Capabilities Needed*

The capabilities needed to answer the software challenges outlined above are aimed at positioning the DoD as an effective buyer of software, ensuring the availability of software support systems that are adaptable to each project, providing for the easy and trusted reuse of existing software, introducing a variety of rapid and cost effective software production and support techniques, and assessing and reducing the risks inherent in the development of new software systems. These capabilities include methods, techniques, and tools to:

- Improve the software creation and evolutionary support processes with software-first design, specification and documentation methods.
- Reduce the cost of software development through the development and application of accessible, trusted, reusable software components.
- Provide for routine use of automation in the software creation and evolutionary support process.
- Reduce the risks inherent in the development of mission critical software by applying appropriate software standards and procedures.
- Demonstrate the transition of new technology to practice on DoD MCCR systems and use the feedback from lessons learned to refine the technology.

## SECTION 3 - TECHNICAL STRATEGY

Severe software problems are not unique to a single Service or Defense Agency. Similar software problems have been experienced by every Defense Department component. These common problems deserve a common solution. The STARS program will concentrate resources to develop and make available solutions to support the entire DoD.

The STARS program will not supplant existing programs in the area of software engineering. Rather, STARS will supplement other efforts with a focused and compatible program. Related Service and Agency software programs must continue as they directly complement STARS in the near term and, in the long term, will provide the follow-on and continuity. The thrust of STARS is to develop and productize software solutions to be used by the defense industrial base. STARS will be successful when the software technology base supporting DoD contractors and the DoD has been raised and the process of maintaining and further improving that technology base has been commercialized by industry.

### *3.1 Leverage Approach [R]*

The DoD requires quality products to support efforts to develop and maintain operational MCCR systems. "Quality products" means that the tools and procedures are not only feasible and applicable, but acceptable for use (i.e., cost-effective, adaptable, reliable, maintainable, and reusable). Sometimes the term "production quality" is used to describe these attributes.

There is an important distinction between promoting technology and commercialization of software support systems that meet the varied and specific needs of the DoD. The STARS approach addresses several facets of the software problem: basic technology, engineering the technology into products, and technology insertion. The people and the organizations who develop and maintain DoD systems have also been considered because products alone fall short as a total solution.

Technology transition and insertion do not occur naturally. Although Defense MCCR systems are built by the U.S. weapon systems industry, the defense market alone is not sufficient to stimulate the needed change. We must foster change that creates competitive advantage for industry in the commercial sector as well. For this reason, the STARS program has been focused to provide U.S. industry a leadership role in improving the defense software development process. The STARS strategy includes explicit linkages to commercial opportunities through emphasis on commercial and government software standards.

The STARS strategy extends deeply into the DoD system acquisition process. The focus will be on continuous relationships between the user community and the development organizations, to produce an end product that meets the needs of the users. Elimination of ineffective development steps and documentation will help motivate the industry to use STARS defined procedures and increase overall quality. New definitions of data rights and contractual obligations will further motivate industry in the direction of improved productivity.

### *3.2 Productivity Improving Opportunities [R]*

STARS will develop and demonstrate ways to significantly increase software productivity over the next 5 years. The DoD will need major increases in software productivity, compared with today's baseline, to get affordability and capability improvements at the same time. A high, near-term

productivity objective also serves as a convenient criterion against which to evaluate the potential contributions of candidate projects and with which to focus the STARS program.

STARS will seek major and early productivity achievements by exploiting opportunities from several different sources. These are: the software development language, the development tools and environments, and software reuse technology. There is a strong relationship between these opportunities, and the STARS program will provide a coordinated approach to realizing the productivity improvements.

Concurrent advances along these dimensions will yield a compound productivity improvement that is significant, well understood, and easily observable in terms of total cost and elapsed time. There is no need or plan for detailed measurements for each STARS program element.

### **3.2.1 Ada - The Language of Choice [R]**

The Ada language provides powerful features that contribute to new levels of productivity in the design, programming and maintenance phases. Although the language is only now beginning to get wide use, reviews [10, 11] of early Ada products provide encouraging evidence that Ada will achieve the productivity improvement sought by the Ada language designers. These results have been achieved generally by people without prior Ada experience using early compilers on conventional machines, so even greater improvement factors are reasonable to expect.

Ada provides an integrated collection of powerful, advanced features (typing, encapsulation, generics, exception handling, tasking, packages, specifications, etc.) selected for the development and maintenance of the software typically found in MCCR systems. Ada provides a well controlled, standard language that permits high degree of source-level portability between validated compilers and provides opportunities to isolate machine and operating system dependencies to achieve a high degree of machine independence.

The Ada language brings with it several technologies that reduce the cost of implementing software. Beyond the clear impact of software reuse, Ada supports many design techniques that reduce the risks of software development. Comparison of Ada code to previous versions in other languages has provided evidence for expecting programs written in Ada to be smaller than the same function in another language [12].

The goals of the STARS program -- increased productivity and increased MCCR software quality -- were fundamental to the design goals of the Ada language. Thus it is natural that the STARS program capitalize on the base provided by Ada activities. Ada can be used to capture the results of the STARS program and share them throughout the DoD software community. Ada will provide a common medium through which different software technology advancements can be consolidated and focused on the problem of increasing the productivity of the MCCR software production process and the quality of the resulting MCCR software.

### **3.2.2 Labor Saving Tools [R]**

The software community has recognized the opportunity to increase productivity by capitalizing the programming force with tools and special architecture software development machines. The programmer support capability possible for a given cost has increased steadily over the years and is predicted to continue to increase [13].

In the labor saving area of opportunity, repository technology to facilitate sharing, transfer and control of software during development, maintenance and reuse will also be developed.

Efforts to reduce the overhead of documentation and project management will also be undertaken, but are not expected to provide comparable productivity contributions. Much more powerful opportunities seem to lie in the direction of changing the software engineering process so that these former, human-intensive, paper driven controls are no longer required.

### 3.2.3 Software Reuse [R]

Software engineers have long recognized that reusing, rather than rebuilding, would be a powerful software approach with significant cost-benefit opportunities [14]. Unfortunately, the common past practice of using many different languages and language versions, and efforts to increase performance through use of machine and operating system dependent features, have generally discouraged widespread reuse. Given the past software base, reinvention has generally been cheaper than reuse. The Ada language with a consistent application of design and coding practices that improve readability, isolate machine and operating system dependencies, and encourage Ada source-level portability should provide a major new opportunity for mission application software reuse. STARS will seek to develop a sizable amount of software in ways that will encourage and foster reuse.

Reusability opportunities will be exploited by STARS at several levels. As a first step, having a large quantity of software covering many different application domains in a common language, with clearly isolated and well documented treatment of machine and operating system dependencies would provide a new opportunity to exploit reusability. As a second step, unification of functional interface standards would introduce additional opportunities for reuse. For example, subsystem components could be more easily separated for reuse. Also, a new and more specific taxonomy of software could be developed to facilitate description, cataloging and retrieval of software components. Automated tools to reconstruct (to the extent possible) requirements level, functional-level and specification-level descriptions by processing source, object, and command language code would further facilitate reuse. At another level, approaches that would reuse trusted subsystems components, their designs and specifications, and their formal proofs need to be explored. There are then several different opportunities for reuse by the software engineer at subsystem, component, package, and subpackage granularities. STARS will exploit reuse at all levels that provide cost-benefit returns.

This strategic objective may be referred to as institutionalizing the reuse of software parts. To achieve it, the STARS program will remove the inhibitors to reuse. These inhibitors are summarized in Table 3-1.

INHIBITOR	STARS SOLUTION
<b>LANGUAGE</b> Non-portable programming languages	Ada Machine-independent programming Portability guidelines
<b>INTERFACES</b> Non-standard interfaces between parts	Commercial interface standards Application blueprints
<b>DEVELOPMENT PROCESS</b> Specification-driven process providing poor user feedback	Prototype-driven process by user feedback System Architect Organization Chief Programmer Team
<b>SOFTWARE PARTS</b> Unavailability of parts	High-function parts library Search/retrieval methods Parts
<b>TOOLS</b> Reuse tools unavailable	Tools for: Parts generalization Parts evaluation Parts customization System tradeoff analysis
<b>ACQUISITION METHODS</b> Discourage reuse of software parts	Recommendations for changes to acquisition process

Table 3-1: Inhibitors to Software Reuse

### 3.3 *Software-First Processes [R]*

Traditional system acquisition processes within the Department of Defense (DoD) have tended to emphasize hardware over software. Hardware has always been part of systems acquisition whereas software has only recently gained a position of prominence. As a result, acquisition processes have been developed, and evolved, with hardware primarily in mind. Software has too often been viewed as an "add-on" to be acquired after the hardware is specified or obtained, despite that computing hardware is a steadily declining portion of system costs.

The acquisition process has been further complicated by a common practice of considering hardware to include a run-time support system upon which application-dependent software must execute. In keeping with the hardware first nature of traditional acquisition processes, the run-time support system is usually specified, and sometimes fully acquired, before the application-related software issues are addressed.

High levels of software cost reduction are believed possible by addressing software first, rather than hardware, in the acquisition of automated MCCR systems. Considering software-first can lead to a much richer set of possibilities for software reuse. Software-first can provide more extensive and easier determination of user requirements before the constraints imposed by the hardware complicate the problem. Considering software-first will also allow the early determination of the software's general structure and will therefore help specify hardware facilities (such as distribution, parallelism, etc.) needed to achieve desired performance levels.

The STARS program will promote processes that support a software-first approach to systems acquisition. The processes are characterized by the following attributes that:

- Provides for early and frequent reassessment of requirements, by end users. This attribute implies a prototype driven process which is directed by user evaluation and feedback.
- Allows and encourages the use of prototyping for requirements formulation, decision exploration, and system evolution.
- Promotes prototyping and development of alternative solutions, which will build a repository of potential solution components. The repository will also contain reused components from other projects.
- Emphasizes assembling systems by selection from alternatives rather than by construction from scratch. The assemblage will follow application "blueprints" which are agreed upon early in the acquisition process.
- Provides a "bi-modal" process by which a system can be developed through the free intermixture of bottom-up and top-down processes.
- Freezes final requirements after most of the software has been developed.
- Provides supporting tools to compose new systems from reusable software and to construct capability from existing code.
- Incorporates both formal and computer aided analyses.
- Delays final selection of the operational hardware after most of the software has been developed, and concurrent with the integration of the system.

Processes with these characteristics are fundamentally different from current requirements-driven software processes that lead to the development of overly detailed specifications before the fundamental problems concerning a software system are identified and understood.

Software-first processes require that the implementation language be machine-independent. This, in turn, requires a strict and enforceable control of the language definitions and implementation. The DoD common high order language, Ada, defined by the Ada language standard [15] and controlled through the compiler validation facility, is the most uniformly implemented language ever produced. Ada provides the best opportunity to support a software-first technology today.

The STARS program will emphasize use of an adaptable (reusable) software technology process to replace the present life cycle software model. Initial efforts will be compared with the life cycle phases defined by traditional "waterfall" models such as MIL-STD-2167 [16]. Attention will, however, be given to an Ada-focused alternative model supporting an evolutionary development process. The STARS approach will make maximum use of Ada for specification, design, implementation, execution, and evaluation of MCCR software. Support tools will also be implemented in and for Ada.

Figure 3-1 present a graphic example of a software-first life cycle.

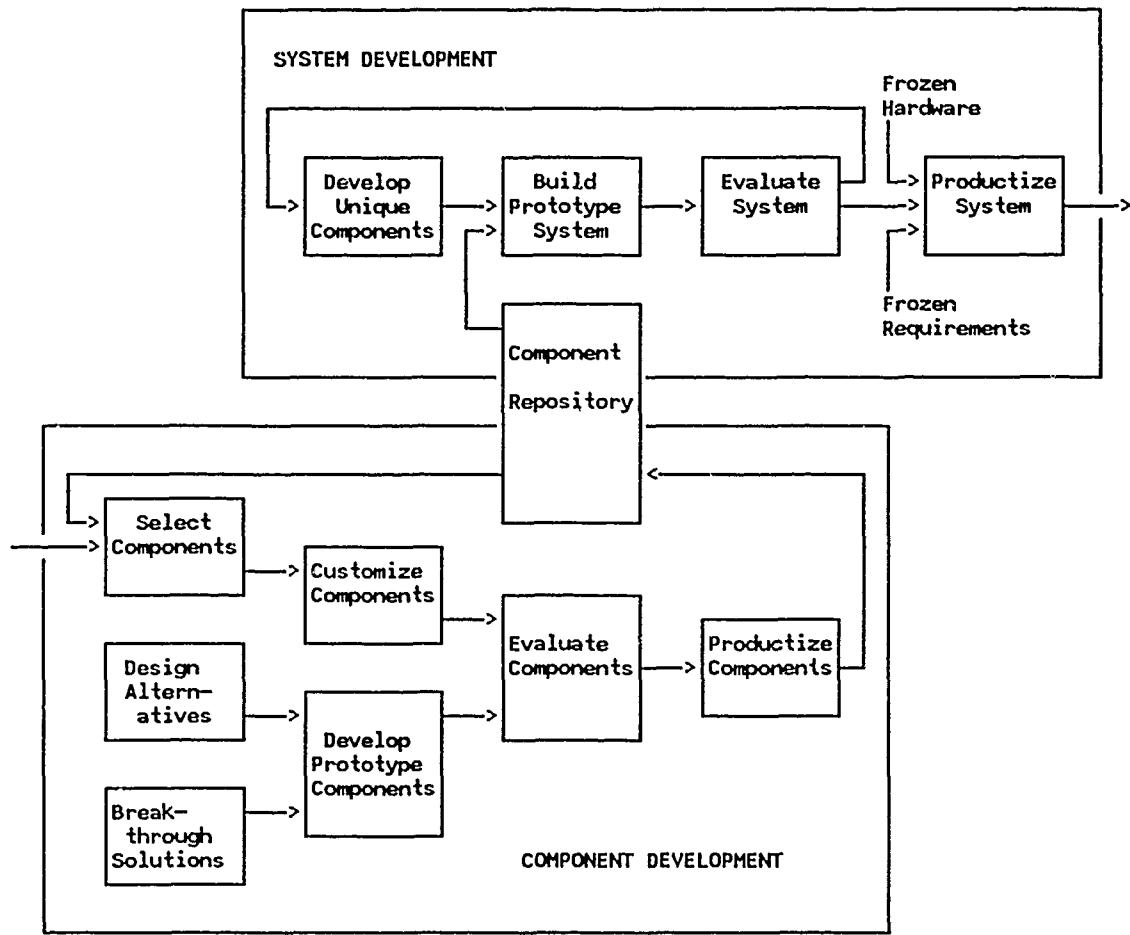


Figure 3-1: Software-First Life Cycle

### 3.4 Standardization Support

The problems of software portability and communication within and across computing architectures have existed in the information processing community from the beginning. Industry and government have developed a comprehensive suite of over 1,000 standards that address software related issues [17]. Employment of technically and functionally appropriate standards has several benefits. First, standards allow the establishment of consistent functional interfaces that support well-layered design and development of components. Second, standards facilitate the reuse of properly designed software components across application domains. Finally, standards capture the fruits of an extensive engineering effort already undertaken by industry and government, thus avoiding replication and re-invention.

STARS will therefore foster the use and development of government and commercial information processing standards as an important part of the STARS program. The goal of the STARS initiative will be to unify a consistent set of standards through bindings and implementations in and for Ada. The fact that STARS will encourage company sponsorship of STARS funded efforts before accredited commercial standards bodies is a key aspect of the STARS leverage strategy for in stimulating commercialization of STARS initiatives and fostering a new private sector markets for the software industry.

### *3.5 Relationships to Other Programs*

Since the initial definition of the STARS program in 1983, several other programs in software have started within and outside the Government. Collectively, they provide a context for consolidating STARS program efforts. The Software Engineering Institute (SEI) has the role of transitioning advanced software technology into widespread practice within the DoD software community. The Defense Advanced Research Projects Agency (DARPA) has developed plans for rapidly capitalizing on advanced software technology at the edge of the current technology horizon. Several programs such as the Ada Joint Program Office (AJPO), World Wide Military Command and Control System modernization (WIS) and the Strategic Defense Initiative (SDI) seek to capitalize upon more well-developed technology to achieve immediate gains.

The STARS program will coordinate and integrate STARS activities with those of other programs. The coordination and integration with respect to the Service programs, in particular, are defined in the STARS program Management Plan [4]. STARS program activities will fill the gap between the immediate results achieved within the Service programs and the long-term results hoped for through other DoD programs. The bulk of STARS activities will be oriented towards medium-term results that will deliver a significant productivity increase.

Coordination and integration with SEI will result in the de-emphasis of transition activities within the STARS program. The initial plans for STARS assume that the task of transitioning new technology developed by STARS will be handled by the SEI. The STARS program, however, will retain activities called shadow demonstrations to assess the value of STARS technology in real-world situations. A secondary intent of shadow demonstrations will be to provide guidance for STARS program planning and execution. The data collected through demonstration will provide a pragmatic measure of the program's progress and also help identify high-payoff directions from among alternative activities.

### *3.6 Changes to the STARS Program [R]*

In FY 1986, following major program reviews, the STARS program was redirected. The revised program concepts, while consistent with the long-standing goals of the STARS program, represented a significant restructuring and focusing of the technology and management approach. Prior to the restructuring of the FY 1986 program, STARS documents described the program in terms of eight different technology areas [18]. The previous Applications Specific and Measurements areas were combined and replaced with a new emphasis on shadowing selected, real, mission-critical system developments to provide STARS technology demonstrations. Shadow projects were defined as STARS-funded initiatives to develop operational-quality MCCR products by applying the Ada-based STARS development processes to real, DoD mission critical systems. The former STARS program technology areas called Systems, Methodology, Automated Environments and Man-machine Interface were replaced by a single, focused, software-first technology development thrust. The former Business Practices and Human Resources areas were not funded by STARS and were referred to the Services and the SEI as technology transition activities.

Prior STARS efforts to automate the current state-of-the-art as represented by the Software Engineering Environment (SEE) efforts were referred to the Services as commendable efforts, but inappropriate for STARS funding because they represent a technology approach insufficient to achieve STARS productivity goals. Service cost estimates of the STARS-SEE preliminary design also made the SEE unaffordable within the STARS program as previously planned. As discussed in Sections 4.3.1 and 4.3.2, STARS will include environment development activities, but they will differ in two important ways from previously planned efforts. First, the STARS environments will support a new software engineering approach. Second they will be developed themselves using that approach. This plan assumes that the change will allow the development of several environments for less cost than previously estimated.

Since FY 1986, twenty-five STARS foundation tasks have been completed, five shadow projects have been supported, and three prime contractors were selected. These program elements have followed the management and technical plans as established in 1986.

In FY 1989, there is significant potential for major program changes due to the following issues.

- The STARS program office was transferred to DARPA in 1988.
- Funding for the program is being adjusted to meet the realities of the current DoD environment.
- The initial increment of the prime contractors was completed in 1989, and ideas expressed in the deliverables will be incorporated into the program.

There are technology areas that will not be addressed by the STARS program. Research in very high level languages, knowledge-based systems, artificial intelligence, 5th generation languages, non-Von Neumann architectures, logic programming and symbolic evaluation will not be funded by STARS. Proposals for work in these areas will be referred to DARPA and the services' 6.1 and 6.2 research programs. These advanced concepts, while potentially very promising, are not judged likely to meet the STARS delivery schedule.

## SECTION 4 - TECHNICAL PROGRAM

The STARS program is a software engineering technology development program. The program builds on the 15-year foundation of Defense Department initiatives that began in the early 1970's following publication of a major Air Force study [19] that identified software costs as a major problem for mission-critical systems. In particular, STARS will build on the Ada computer language and related software engineering processes.

### *4.1 Program Structure*

To capture the opportunities of the Department of Defense Ada software engineering initiative, the STARS program is structured with four fundamental and related thrusts:

1. The success of the STARS-developed software engineering process will be measured pragmatically through "shadow" projects that develop real operational mission critical applications in Ada.
2. Software engineering environment development will focus on a software-first approach to system acquisition.
3. The technology development thrust will focus on adaptable and reliable software engineering approaches.
4. To solve fundamental research issues, STARS will pioneer an approach to risk reduction applicable to software-intensive mission critical systems. A software repository will be maintained to facilitate sharing of reusable software.

Each of these thrusts will be described in greater detail in the following subsections. The STARS program is based on industry leadership through a few lead contracts and includes participation of Service Laboratories in research areas and Service Product Divisions in shadow projects.

### *4.2 Shadow Demonstrations [R]*

Shadow demonstrations, using the discipline and processes developed by STARS, will be selected from weapons, command and control, and intelligence areas to provide representative demonstrations for each Service. Shadow projects will provide a pragmatic measure of the STARS program progress in developing and introducing a new software engineering approach, provide realistic and useful feedback to the technology developers, and contribute to the process of technology insertion. The "should-cost" goals for shadow developments will be set by the STARS Director and reduced over the program years to eventually demonstrate the productivity improvement sought. Shadow activities serve to demonstrate the feasibility and value of using STARS-developed software technology on real defense systems software projects. The tools and techniques underlying these demonstrations will be developed as part of the STARS technology development activities. The focus is on the actual demonstrations themselves.

Reusable end-application software system components will be developed in support of these demonstration activities. The development of these components will therefore be driven and their definition will stem from actual projects. Coordination by mission needs among the various

shadowing projects will result in the identification of generic components usable over a variety of projects and application areas.

Most candidates for shadow activities will be selected from system programs nominated by the Services and DoD agencies. The actual programs to be shadowed will be selected in coordination with the STARS Director, the Service or Agency, the System Project Office Director, and industry contractor involved. STARS will fund the shadow projects.

The following criteria are a few of those that will be used to select STARS Shadow demonstration projects. The program should:

- Be in the weapons, command and control or intelligence area.
- Provide potential for software reuse.
- Use a traditional software acquisition approach with a non-Ada implementation.
- Have a projected size of 100K lines of code or more for the non-Ada implementation.
- Commit to demonstrate a STARS-developed software technology.
- Permit portions of the Ada shadow application software to be placed in a STARS repository under appropriate access controls.

### *4.3 Product Development*

Powerful market forces are at play. The sweeping introduction of the personal computer, the mass production of low-cost and powerful commodity software and the emergence of third-party software marketing organizations have changed the computer business in ways to which no organization is immune. We cannot predict the timing, the basis, or the extent of similar forces in the future. We can predict that such phenomena will occur.

In picking a marketplace strategy for a program like STARS there are several dangers. Bucking the tide can be very costly. At the same time, locking in on yesterday's force can miss the next big opportunity. For example, were STARS to lock on to today's powerful commodity software market force as the key to tomorrow's success, STARS could miss the next fundamental new market force. If, for example, the industry provided the ability to adapt software to the unique requirements of each using organization in near real-time to give that organization a competitive edge, a new market with a service orientation could be added to today's commodity oriented market. Adaptability, in fact, is the capability that MCCR mission critical DoD systems need; and, therefore, is the marketplace change that STARS seeks to foster.

The STARS market strategy will include the development of many new products. The products that STARS will develop, while intended to be usable and useful in their own right, are intended more fundamentally to be adaptable and reusable. The STARS product is not the end, but rather the means by which a new process of software engineering will be developed, demonstrated and explained. The STARS process will open new markets and new opportunities for commercialization. That commercialization may be a very different service oriented opportunity that goes beyond the capabilities that are possible solely in a product oriented commodity market.

STARS product development will be done in a way that seeks to take the early cost risk out of a new process, but allows maximum opportunity for industry to extend their efforts. Widespread sharing of the results of work funded by STARS is essential as the catalyst by which a new adaptable technology process will be developed, understood and extended by industry.

The STARS process-oriented and service-based marketplace strategy underlies the choice of principles (Section 4.3.2) and technical guidance (Section 5). The principles and guidance as presented are not final, but should provide a strong signal to the technical reader that something different is going on in STARS. STARS seeks to provide cost leverage in mission application software. The approach should also provide cost leverage in development of the support environments. The program includes the use and demonstration of the new processes in both domains.

### 4.3.1 Technology Integration

The STARS lead contractors [4] will each be responsible for integrating the results of research and technology development efforts (described in Sections 4.4 and 4.5) to produce software engineering support environments specialized for a particular application domain. Each environment should make maximum reuse of common Ada foundation capabilities (described in Sections 4.4.1.1). The responsibility to deliver these software engineering environments is intended to provide the lead contractors a practical focus for their research and technology oversight responsibilities. The software engineering environments are intended to be a demonstration of the software-first software engineering approach they seek to support.

### 4.3.2 MCCR Software Engineering Environments

The STARS prime contractors will develop three prototype software engineering environments for peer review and evaluation. These systems will be built using common, reusable Ada software unified by a consistent set of functional interface standards to provide an adaptable framework for interfacing tools and processes. A configuration control discipline will include tools for the protection of software at different levels of configuration control and for promotion between levels of configuration control. The system will include documentation and specification approaches supported by computer aided processes. A significant and growing set of tools for reliability and adaptability processes will be defined and under development.

By the end of the prime contracts, three software engineering environments will be delivered. These will be adapted to the unique requirements of specific application domains. The design and integration approaches used in the development of these environments should demonstrate a new level of adaptability based on a common, reusable software technology. The time and cost to develop future environments adapted to the special needs of a particular application area should be demonstrably reduced over today's practice. These products are intended to be the processes by which future application systems will be developed and maintained. The adaptable, reliable software-first technology assures that the process is more important than the application product at any time because the process provides the capability to respond on useful time scales to changing threats.

#### 4.3.2.1 Frameworks [R]

A fundamental aspect of a software engineering environment centers on the framework that such an environment provides for supporting and interfacing the tools and capabilities within the engineering process. Several principles will be adopted as technical invariants to guide the software engineering environment developments. As invariant principles, the development environments must:

- Use unmodified commercial operating systems.
- Use a validated Ada compiler.
- Use Ada as a command language through a common command language interpreter.
- Use common command words consistent with Ada in new tools.
- Be machine independent down to a minimal set of well defined, low level interfaces, where implementing bodies will respond to the specific environment.
- Interface to external facilities, when used, through standard interfaces for virtual terminal, network, data base, graphics and file naming conventions.
- Be hostable on any system sufficient to host a production Ada compiler.
- Allow hosting multiple environments on the same hardware without compromising information or reporting to a central control.

- Assure that management tools impose no additional burden on the programmer or slow the development process.
- Keep the environment overhead to not more than a factor of 2 over doing a like function with a manufacturer's development environment.
- Use existing, off-the-shelf software whenever possible, both within the environment itself and in applications being developed under the control of the environment.

#### *4.3.2.2 Software Process Control [R]*

An automated control process capability with clear levels of control (such as development, test and operation), and with a multi-level trusted promotion process will provide a fundamental contribution to software reliability. These capabilities must:

- Support the reuse of existing software by introduction of a reuse taxonomy and process extending from requirements to code and from code back to requirements.
- Provide a mechanism to search for and access reusable software components.
- Provide life cycle support of software and its associated documentation including multi-level trusted configuration control of software and document versions.
- Use the Standard Generalized Mark-up Language (SGML) for all software documentation preparation.
- Be developed in an evolutionary manner, supporting extensibility of capabilities throughout the environment's life.

#### *4.3.2.3 Database [R]*

Library management, configuration management, and repository management require complex data systems. To achieve the level of integration proposed in the environment, a major emphasis must be devoted to centralizing these requirements in a logical structure. The database must:

- Build upon the underlying file system.
- Provide for the definition of object types and classes.
- Support the relocation of information between geographically distributed systems.
- Support outside data paths to allow relocation of information between distributed subsystems.
- Provide tools for protection of and promotion between configuration control levels.
- Use an abstract Ada data dictionary for integration.
- Support the registration of abstract document types that would be available over data communication networks from a repository coupled with a generic editing capability specialized by the registered type.
- Demonstrate access to a remote data dictionary that could be part of the repository system.

### *4.4 Technology Development*

The Shadow activities discussed in the previous sub-section provide direct support for moving modern software technology into the DoD software community. Improvements can be obtained by developing and making available technology that is currently within the state of the art. Other improvements require investigations into fundamental issues. The STARS program technology development, productization and research activities will provide future capabilities to be transferred

into practice. This section discusses STARS technology development efforts that focus on adaptable and reliable software technologies.

#### **4.4.1 Adaptable Software Technology**

The STARS program will seek an adaptable software technology by developing a critical mass of foundation capabilities in machine-independent Ada as the core of a reusable technology. Integrability and reusability will be enhanced through a strong focus on commercial standards (e.g., those of IEEE, ANSI, and ISO standards bodies) at the functional interface levels. A major thrust will be undertaken to provide a trusted repository to describe, locate and access end-application and tool software (within the constraints of export controls and security concerns). An Ada-based, bimodal technology for design-by-successive-refinement will be developed to support software reusability concepts. Along with the process, there will be an emphasis on eliminating the overhead burden of duplicative and often redundant documentation requirements.

##### ***4.4.1.1 Foundation Ada Capabilities***

A prerequisite to an adaptable software engineering process is a critical mass of software that is available, organized and structured with a view to adaptability. Because Ada is relatively new and because widespread system development in Ada is just beginning, the opportunity exists to achieve considerable economies by focusing on the common software found in almost every system. STARS will seek to develop such common software in a way that facilitates and encourages reuse. This common software, called foundation capabilities, includes work in areas such as:

- Command Languages
- Text Processing
- Graphics Protocols
- Operating System Capabilities
- Data Base Tools
- Network Protocols
- Planning and Optimization Aids
- Design and Analysis Tools

STARS will build on the efforts of many government contractors whose Ada products have been delivered to the Ada repository available on SIMTEL20 through MILNET. The following list represents of some of the products that have already been delivered or are under contract.

- Symbolic Debugger
- Automatic Path Analyzer
- Metric Instrumentation
- Source Formatter
- Path Analyzer
- Statement Profile Report
- Complexity Measures Report
- Cross Reference Package
- Compilation Order Report
- PDL Processor
- Design Requirements Traceability
- Documentation Manager
- Automatic Specification Analysis
- Complexity Measurement Analysis
- Graphics PDL Support
- Standards Checker
- General Management (Costing)
- General Management (Work Flow Control)
- Data Dictionary System
- Program Design Assistant
- Standards Support
- Documentation Reports

- Set, Package, and Subprogram Use Reports

Appendix A provides a brief summary of some of the foundation capabilities that were candidates for development. The foundation work was undertaken concurrently with other STARS efforts. For this reason, the work was competitively contracted prior to the selection and award of the STARS lead contracts. These foundation capabilities will be subsequently modified (adapted) to comply with function interface standards that are discussed in the next section.

#### **4.4.1.2 Standards**

Over 1000 standards have been identified [17] that relate to software processes. STARS will seek to unify a meaningful subset of these standards through Ada implementation and bindings. The following list illustrates the kinds of standards that will be treated.

##### **Programming**

- Ada Programming Language
- Ada as a Common Command Language
- Ada as a Common Program Design Language
- Descriptive Intermediate Attribute Notation for Ada (DIANA)

##### **Networks**

- Internet Protocol (IP)
- Transmission Control Protocol (TCP)
- User Datagram Protocol (UDP)
- Teletype Network Protocol (TELNET)
- Internet Control Message Protocol (ICMP)
- Open Systems Interconnect (OSI)

##### **Applications**

- File Transfer Protocol (FTP)
- Trivial File Transfer Protocol (TFTP)
- Simple Mail Transfer Protocol (SMTP)
- Remote Procedure Protocol
- X-Windows

##### **Text Processing**

- Computer Language for the Interchange and Processing of Text (CLIP)
- Standard Generalized Mark-up Language (SGML)
- Page Description Language
- Document Content and Interchange Protocols

##### **Graphics**

- Graphics Kernel Standard (GKS)
- North American Presentation Level Protocol Standard (NAPLPS)
- Programmer's Hierarchical Interchange Graphics Standard (PHIGS)
- Computer Graphics Metafile (CGM)

##### **Database**

- Common Ada Programming Support Environment (APSE) Interface Set (CAIS)
- Structured Query Language (SQL)
- Information Resource Dictionary System (IRDS)
- Distributed Database Protocols
- File Server and Receiver Protocols

##### **Operating Systems**

- Portable Operating Systems (POSIX)

Figure 4-1 illustrates the concept of how commercial interface standards might be used to build generic system components. STARS will use commercial standards to foster such leverage through reuse across functionally diverse applications.

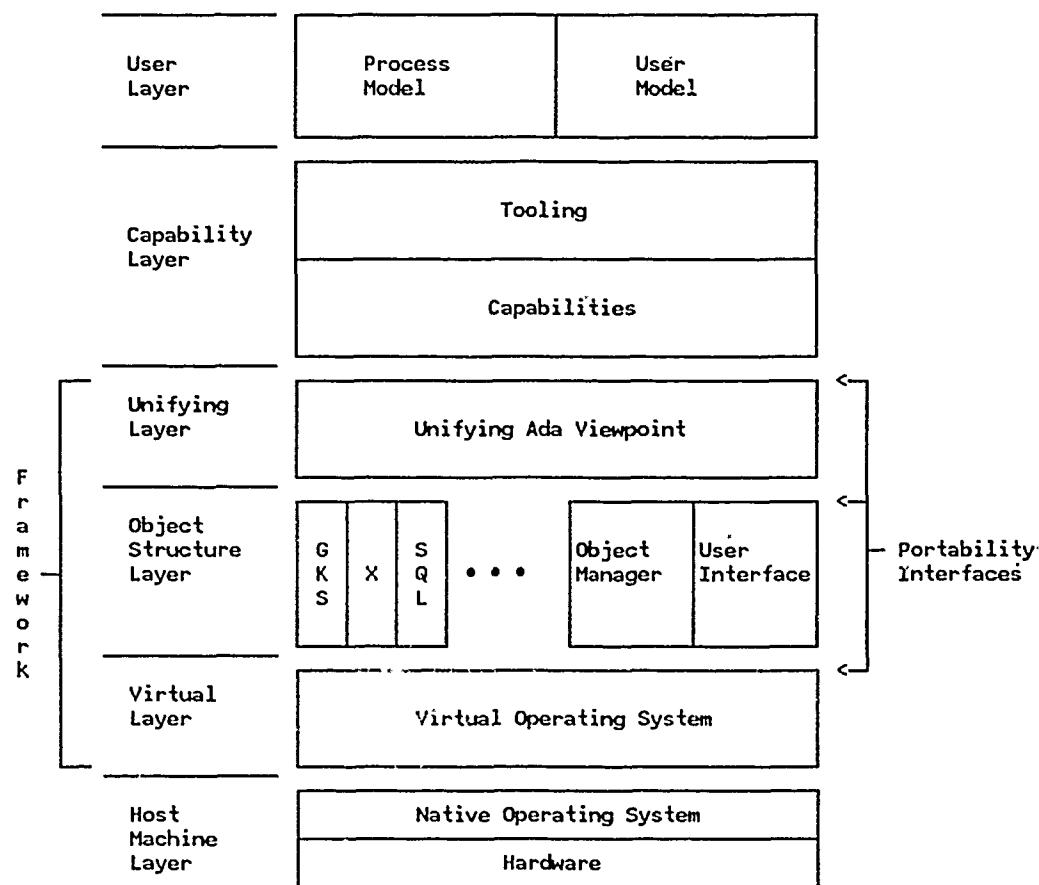


Figure 4-1: Example of Functional Interface Standards

#### 4.4.1.3 Repository

STARS will maintain a high quality on-line, trusted, mandatory-access-controlled software repository with continuously improving human interface capabilities. The repository will be used to make available processes and the tools and tool collections supporting them. Software processes supporting the creation and evolution of software systems can be used equally well to support the development and maintenance of automated environments or end-application systems. General approaches and specific methods will be included in the repository so they can be accessed by the STARS community.

Special attention will be given to processes supporting reusability. In particular, the repository will hold processes, tools and tool collections needed to make use of the repository itself. The repository will provide access to a variety of reusable software system fragments. Assistance will be needed to identify the fragments pertinent to a particular project, evaluations, adapt the chosen fragment as necessary, and integrate them with other (new or reused) fragments. The repository will, therefore, contain and provide access to the selection and assembly processes supporting the use of elements in the repository for the development of automated environments and end-application systems.

#### **4.4.1.4 Design by Successive Refinement**

The package feature of the Ada language supports the separation of the design process from the implementation process. The application of tools that automatically create package body stubs from package specifications allow the Ada compiler to be used as a design tool for checking design completeness and consistency. The capability introduces the opportunity for a process called design by successive refinement in which compilable designs can be ever extended with additional capabilities while simultaneously providing a continuing check of completeness and consistency. This concept applies equally well to functional decomposition and object oriented design.

The concept of design by rapid prototyping is a logical extension of iterative refinement process. Prototyping, as a basic component of the software-first life cycle, is thus well suited for Ada software development.

These capabilities need to be part of the standard design process. Additionally tools need to be developed to support the process beginning with existing code as well as requirements statements.

### **4.4.2 Reliable Software Technology**

Reliability will be founded in computer aided methods, Ada-technology domain extensions to include system views, formal methods for functional correctness (and security), and automated configuration control processes for a new reusable-software evolution concept.

#### **4.4.2.1 Computer Aided Methods**

Several tools are available in the Ada repository on SIMTEL20 to aid in the software development process. Another class of tools that could contribute to more reliable software in the near-term are known as validation capabilities as represented by the Stanford University, ANNA (ANnotated Ada) specification and validation system. Capabilities of this software need to be identified and integrated into the system environment.

#### **4.4.2.2 Domain Integration**

The classical software engineering process has been generally defined in terms of a series of activities proceeding sequentially from requirements, to specifications, to design, to coding, to testing, and finally to maintenance. The Ada language has tightly coupled the design and coding steps through the Ada package feature. Many compiler builders use the DIANA (Descriptive Intermediate Attribute Notation for Ada) internal representation of the Ada language and a technology called directed attribute trees to describe relationships. The directed attribute tree technology is supported by the Ada compiler industry base. It includes methods similar to those used by the university and academic research community to relate policy statements to policy models, and top level specifications and detailed design specifications to formal verification techniques. This observation has prompted the proposal to explore the integration of Ada package designs, upward to the top level specification and requirements domains by involving relationships at the internal representation level. The new industrial base would be used in developing these concepts.

There are some fundamental new research issues associated with this concept, but a demonstration of feasibility could open a powerful new approach to integrating the specification and design process that would make a major contribution to software reliability.

The extent to which Ada has been useful in the STARS/VHSIC sponsored effort to describe hardware design, as a first step toward an integrated system design process addressing both software and hardware, suggests that the design process could be extended downward to include hardware and systems domains in an integrated way. Such a downward unification would also contribute to system reliability. These sorts of opportunities will be explored in STARS.

#### 4.4.2.3 *Formal Methods*

Formal mathematical methods have been used to "verify" that two different descriptions (e.g., top level specification and detailed design) of the same system are consistent. Formal methods offer one approach to improve software reliability. Their use is prominent in the development of trusted, multi-level secure operating systems. The methods have also been used for correctness and reliability analysis of some critical, high value MCCR systems.

Today's methods are, however, not affordable for widespread application. As one progresses from the functional requirement, to top level specification, to detail design, to code, and to execution, the process of "verifying" full and exact correspondence between successive views of the same system becomes increasingly difficult. The research needed to develop and productize formal methods for widespread application would exceed current STARS funding levels. STARS will maintain close coordination with those DoD Agencies with primary responsibilities acquiring such capabilities. At a minimum the Ada Joint Program Office effort to develop formal semantics for Ada must continue with U.S. participation. Some STARS projects in these areas are planned within available funding limitations. STARS may fund some development of Boyer-Moore theorem provers in Ada and for Ada. These activities will be coordinated with those of the National Computer Security Center and organizations using formal methods for functional correctness analyses.

### 4.5 *Breakthrough Initiatives [R]*

Experience in shadow demonstrations, environment applications and technology development will provide the basis for annual industry briefings, at which fundamental research issues will be identified for creative industry research. Pre-prototype projects will provide a new model for dealing with high-risk MCCR acquisition issues.

The present, requirements-directed approaches have historically focused on the development of elaborate specifications before the fundamental problems are identified and understood. This leads to partially or completely ineffective software and systems or runaway development costs. This "waterfall" approach guarantees a slim chance of meeting the initial system requirements.

An alternative approach to risk reduction is to cultivate multiple paths to the desired system. As the engineering development activities proceed, some solutions will produce changes to the baseline system with a reasonable effort, but other solutions will require additional technology tasks. In the same manner, some technology activities will flow directly back to the engineering development, while other technologies will require breakthrough developments to establish potential use in the project.

This method is synergistic with the software-first life cycle, as alternative solutions are developed and evaluated throughout the program. A portion of the development budget is continuously spent looking for breakthroughs, and risk is accepted throughout the life cycle. There is no great technical wall that blocks progress or delays development.

The general model for breakthrough initiatives is diagrammed in Figure 4-2.

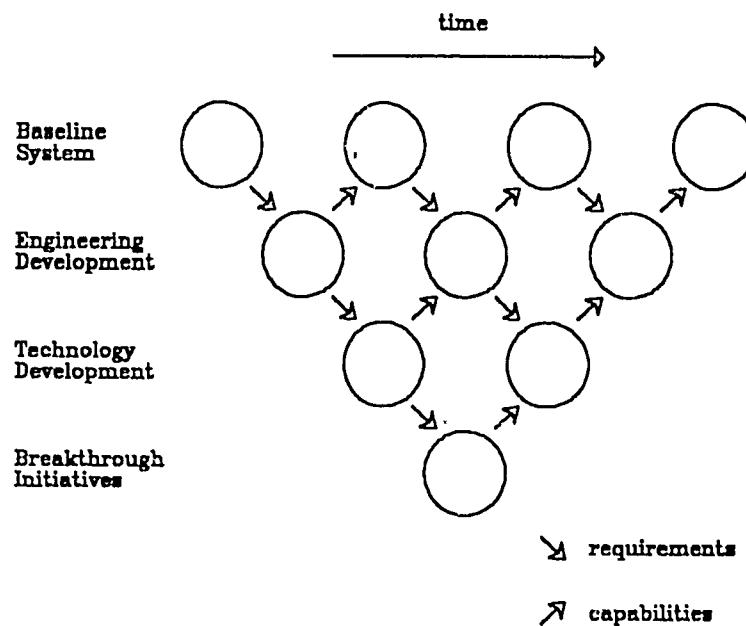


Figure 4-2: Breakthrough Initiatives

## SECTION 5 - TECHNICAL GUIDANCE [R]

The following general technical guidance will apply to all STARS activities:

- The STARS program will place strong emphasis on the development of compilable specifications, designs and executable code in Ada. To the maximum extent feasible, all code will be in machine independent Ada. The syntax and semantics of Ada will be used to the maximum in all supporting and descriptive work. Design will be developed using Ada as the Program Design Language. The designs should be compilable using any validated Ada compiler. Design information should be contained in compilable and executable Ada rather than textual comments. No comment information will be included in code products which are duplicated in processable Ada.
- All deliverable code will be compiled and executed by Ada compilers operating on at least two different instruction set architectures to encourage developers to include machine independence considerations from the design forward.
- Compliance with DoD Directive 5000.31 or equivalent policy statements on the use of Ada is mandatory unless justified and explicitly approved in writing. Failure to comply will result in the decrement of a participating executing agency's following year funding by the dollar amount of projects not in compliance.
- DoD, national, and international standards will be used and supported. Emerging standards will be monitored and incorporated when possible.
- Ada will be the "command language" for interfacing between programs and for interfacing to the menus, and displays at the man-machine interface. Opportunities for using window interfaces should be explored.
- Design should proceed by refinement and modification of product code. The design statements should be suitable for maintenance throughout the software life so that the final code contains the accurate design information.
- The description of systems to be verified will be in an Ada form. Such restrictions and modifications to the exact language as are required for the particular technique under investigation will be kept to a minimum and explicitly justified.
- Maximum use will be made of automated tools. Because of the language commonality (i.e., all tools and environments will be in and for Ada) the construction and sharing of tools will be facilitated.
- As a general practice, studies and activities that do not lead to, or accompany, the delivery of executable products in and for the Ada technology, shall not be funded.
- All proposals for contract or government internal work will be judged on the technical merits of the write-up that should include, but not be limited to:
  - Statement of problem or opportunity
  - Identification of prior and related work
  - Identification of potential applications and users
  - Summary of pertinent literature
  - Summary of applicable standards
  - Nature of expected contributions or benefits
  - Basis for expected contributions

- Cost estimates
- Basis for cost estimates
- Demonstration that work has not been done
- Documentation standards must be defined early in the contract performance period. The intent is to share the documents in the same fashion as the Ada code. To facilitate this sharing, the documents must be delivered in a common electronic form that has reasonable support within the industry.
- All deliveries will be in electronic and hardcopy forms. General access computer repositories will be used to store the deliveries.
- The intent is to place all delivered work into the public domain. Appropriate Federal Acquisition Regulations (FAR) will be specified in the contracts to cover the data rights issues.
- A peer review process will be used to ensure quality and usefulness of deliverables. The peer review process must be defined early in the performance period. The use of electronic mail networks is encouraged.

## SECTION 6 - PRODUCTS AND MILESTONES

The STARS program is structured to reach several milestones and produce specific products. They include:

### *6.1 Shadow MCCR Systems [R]*

Twelve MCCR Shadow demonstrations were conducted between FY 1986 and FY 1988, with the aid of STARS funding. These demonstrations resulted in delivery of operational, mission quality Ada software suitable for use by the Services. Additional information on these projects is in Appendix B.

### *6.2 Software Development Environments [R]*

Several software engineering development and maintenance environments, each specialized for a particular applications domain, will be delivered under the prime contracts. A family of software engineering tools to support a reliable and adaptable software technology consistent with an evolutionary software-first approach to systems acquisition will be delivered and refined during the STARS program. Additional information on the environments is presented in Appendix C and the Competing Prime Contractors Consolidated Plan [20].

### *6.3 Technology Development [R]*

In the summer of 1984, the WIS Joint Program Office contracted through Naval Ocean System Center (NOSC) for the development of Ada software. Proposals were sought from industry against 57 categories of software including tools and applications. The work has lead to the delivery of software, some of which is listed in Section 4.4.1.1.

STARS has built on the successful NOSC initiative and has extended the Ada tools currently available at the Ada repository on SIMTEL20, by developing a significant quantity of reusable foundation software capabilities in Ada during FY 1986, 1987 and 1988. The foundation capabilities are discussed in Appendix A.

STARS will seek to unify, using Ada, a meaningful set (e.g., those covering operating systems, data base, graphics, network, documentation, etc.) of functional interface standards as a means of establishing a reusable software technology. This effort will continue throughout the STARS program but will be sufficiently mature by FY 1989 to begin adapting the foundation capabilities to these interface standards.

STARS will develop and use improved repository capabilities accessible over MILNET to store and make available the STARS products and technology. In FY 1986, the Ada repository at SIMTEL20 was used. In FY 1987, the Naval Research Laboratory (NRL) established a repository for deliveries under the STARS foundation contracts. This repository will remain operational through the end of the STARS program.

In FY 1988, two STARS prime contractors, IBM and Boeing, established two separate STARS repositories. Initially they contained the NOSC and foundation repository material. They are being

used to hold all the material delivered under the prime contracts. These repositories will be enlarged and improved throughout the program. Following the STARS program, they will become the basis for a robust DoD repository. The prime contractors have defined a road map to products in their consolidated plan [nn].

## *6.4 Fundamental Research [R]*

STARS will seek to pioneer and demonstrate a new approach for dealing with state-of-the-art software technology issues in MCCR systems acquisition. The approach will result in several briefings for industry each year on fundamental issues needing resolution. Additionally a rapid response mechanism will be established to fund creative solution proposals from industry.

Rather than attempting to write detailed specifications as the result of the systems engineering process, STARS will instead use pre-prototype and prototype activities leading to executable, feasibility demonstrations of solution approaches. These demonstrations will provide the basis for credible specifications in support of process and product development, that in turn support mission applications or environment integration.

## SECTION 7 - REFERENCES

- [1] CHARTER, Software Technology for Adaptable, Reliable Systems (STARS), Office of the Under Secretary of Defense, R. D. DeLauer, 1 November 1984.
- [2] Memorandum for Joint Logistics Commanders, "Staffing and Management of DoD Software Programs," Deputy Secretary of Defense, W. H. Taft, IV; 12 August 1985.
- [3] STARS Program Requirements Document, Prepared by Joint Logistics Commanders - Computer Resources Management, Recommended to Deputy Secretary of Defense, January 1986.
- [4] STARS Program Management Plan, Submitted by STARS Director, STARS Joint Program Office, May 1986.
- [5] "DoD Computing Activities and Programs - 1985 Specific Market Study," Market Planning Reference Publications, The Requirements Committee, Government Division, Electronic Industries Association (EIA), December 1985.
- [6] US Army Science Advisory Board (SAB), 1983 Summer Study on Acquiring Army Software, December 1983, Department of the Army, Assistant Secretary of the Army, Research, Development, and Acquisition, Washington, D.C. 20310.
- [7] USAF Science Advisory Board Ad Hoc Committee on the High Cost and Risk of Mission Critical Software, December 1983.
- [8] Defense Science Board (DSB) Task Force on Software, Chartered by the USDRE, 2 November 1984. (The Task Force Chairman, Dr. Fred Brooks, presented a preview of DSB finding to DUSD (R&AT) in April 1986).
- [9] "An Assessment of the STARS Program September-October 1985," prepared for the USDR&E by the Institute for Defense Analyses, in three volumes, December 1985.
- [10] Greene, J. S. Jr., "The National Computer Security Center", SIGNAL, Journal of the Armed Forces Communications and Electronic Association, September 1985.
- [11] Schill, J., Smeaton, R., and Jackman, R., The Conversion of Command & Control Software to Ada: Experiences and Lessons Learned, Ada Letters, SigAda Special Interest ACM, Vol IV, Issue 4, January-February 1985.
- [12] Whitaker, W.A., Keynote Address, Second International Ada Applications and Environments Conference, sponsored by the Institute of Electrical and Electronics Engineers, Miami, Florida, 8-10 April 1986.
- [13] Boehm, B., Software Engineering Economics, Prentice-Hall, NY, 1981.
- [14] Boehm, B. and Standish, "Software Technology in the 1990's Using an Evolutionary Paradigm", IEEE COMPUTER, Vol 16, No. 11, pp 30-38, November 1983.
- [15] Reference Manual for the Ada Language, ANSI/MIL-STD-1815A, Department of Defense, February 17, 1983.
- [16] Defense System Software Development Military Standard, DoD-STD-2167, Department of Defense, 4 June 1985.

- [17] Nash, S. H., Redwine, S. T, Jr., "Information Interface Related Standards, Guidelines, and Recommended Practices", IDA Paper P-1842, Institute for Defense Analyses, 1801 N. Beauregard St., Alexandria, VA 22311, July 1985.
- [18] Strategy for the Software Initiative (STARS), DUSD (R&AT)/Dir CSS, March 1983.
- [19] "Command Control Information Processing for the 1980s", CCIP-85, Published in 12 volumes, United States Air Force, September 1972.
- [20] IBM Systems Integration Division, "Consolidated Technical Development Plan for the Software Technology for Adaptable, Reliable Systems (STARS) Competing Prime Contractors", CDRL Sequence No. 0070, November 11, 1988.

## SECTION 8 - ACRONYMS [R]

*Acronym*   *Meaning*

AJPO	Ada Joint Program Office
ANNA	Annotated Ada
ANSI	American National Standards Institute
APSE	Ada Programming Support Environment
CALS	Computer-Aided Acquisition and Logistics Support
CDRL	Contract Data Requirements List
DoD	(United States) Department of Defense
DARPA	Defense Advanced Research Projects Agency
DIANA	Descriptive Intermediate Attributed Notation for Ada
ERA	Entity-Relationship-Attribute
FAR	Federal Acquisition Regulations
IBM	International Business Machines
IEEE	Institute of Electrical and Electronic Engineers
IRDS	Information Resource Dictionary System
JLC	Joint Logistics Commanders
MCCR	Mission Critical Computer Resource
NOSC	Naval Ocean System Center
NRL	Naval Research Laboratory
OSI	Open Systems Interconnect
PDL	Program Design Language
PMP	(STARS) Program Management Plan
POSIX	Portable Operating Systems
SDI	Strategic Defense Initiative
SEE	Software Engineering Environment
SEI	Software Engineering Institute
SGML	Standard Generalized Markup Language
SQL	Structured Query Language

**STARS** Software Technology for Adaptable, Reliable Systems  
**TPP** (STARS) Technical Program Plan  
**USDRE** Under Secretary of Defense for Research and Engineering  
**WIS** WWMCCS Information System  
**WWMCCS** World Wide Military Command and Control System  
**WYSIWYG** What-you-see-is-what-you-get

# Appendix A. FOUNDATION PROGRAM

Foundation capabilities developed in FY 1986 through FY 1988 provided general capabilities that will support future STARS program efforts. These capabilities are applicable to the preparation of both automated environments and end-application software systems.

## *A.1 General Fragment Capabilities*

The general components, applicable to the preparation of both automated environments and end-application systems, fall into six categories:

- Command languages.
- Text processing.
- Database.
- Operating systems.
- Graphics.
- Network protocols.

These are discussed in the following subsections.

### **A.1.1 Command Languages**

A command language provides a communication interface between a computer system and its users. It must enable each user to solve problems on a semantic level appropriate to those problems. In most DoD software developments, there are usually DoD developer/users of varying levels of expertise who must communicate with the host development system and with each other. Their interactions need to be facilitated by a common, uniform, consistent interface.

Activities in this category will provide prototypes of a variety of command languages and command language support facilities. The prototypes should provide multiple options for program control and allow the user to manipulate the system, control the session and data, manage processes and resources, and package primitive commands. These prototypes will allow for the development of uniform, consistent man/machine interfaces. The separable functions of the command language capabilities should be partitioned into packages to facilitate maintenance.

Specific activities falling within this area could include:

- Emulation of a form-terminal on a page-terminal.
- Command string parser/interpreters.
- Command language implementations conforming to existing standards.
- Form-based implementations of standard command languages.
- Form and menu generation tools for bitmap-, page- and forms-terminals.
- Guidelines for interface design.
- Interface to existing components, such as "window" systems.

### A.1.2 Document/Text Preparation

The objectives for these activities should be to develop a prototype Ada-based "document management" system with capabilities for, but not limited to, word processing, providing output with multiple-type fonts, user-oriented "help" messages tailored to the operations being performed, and user history and expertise. Design issues to be resolved should include the provision for compatibility with multiple subsystems that may not have been completely defined/developed, the ability to use a variety of input/output devices, the capability of using textual syntax/semantics to provide assistance in document preparation.

Prototypes here should include those for demonstration of (1) automatic generation of text-based systems, (2) user interfaces for text processing, and (3) integrated packages of writer's workbench tools. Emphasis should be towards "what you see is what you get (WYSIWYG)" systems which permit multiple views. Preparation of electronic documents and of reasonable quality paper documents should be prototyped first. All these prototypes should be extensible. These systems should assume and plan for advanced input/output technologies, should be input/output device independent, and should use virtual input/output devices. They may potentially use and integrate with the latest in desk top publishing.

Specific projects falling within this area could include:

- Text/editor formatter generators.
- Interface modules supporting common document preparation tasks.

### A.1.3 Database Support

The objectives for these activities should include prototype demonstration of potentially standard Ada interface for a portable commercial off-the-shelf DBMS. The prototype Ada-DBMS might provide capabilities for data base definition, including the provision of multiple views (i.e., relational, network hierarchical) of the same database, efficient retrieval and update of individual objects or groups of logically related database objects, authorization control to the field or attribute level, capable of expansion to provide multi-level security control (e.g., prohibiting a query that is authorized to access aggregated data at a given classification from generating secondary accesses to data objects at a higher security classification), multiple interfaces, including programming and query languages, screen-oriented command language/displays, bulk, load/unload facilities, report writer and application generator, multiple user access with backup and recover; and database administration and control.

Design issues which might be resolved here include support of fully distributed access, including partial or full replication of objects; the ability to incorporate "database machines" or "back-end database processor" technology for retrieval operations, expert system interfaces, verifiable security protection including the ability to operate in a multi-level security environment, efficient processing of multiple views or schema, and hardware independence and portability to include optimization of retrieval strategies.

The following potential database management system tools/components should be considered for prototyping.

- An indexed file access package.
- An entity-relationship-attribute package.
- A concurrency control package.
- A work station relational database system package.
- A multi-user relational database system package.
- A relational database design tool kit.
- A view definition facility.
- A view query and update facility.
- An object manager system.
- An authorization package.
- A database query optimizer/compiler.
- A distributed database access package.
- Distributed database protocol management.

- Database operational tools.
- Database application tools.

Based on criteria which emphasize independence, general understanding of functionality and implementation methods, and immediate usefulness, three components can be identified as candidates for early implementation. indexed file access package, work station relational database system, and database application tools. It appears reasonable to prototype these components in parallel.

#### A.1.4 Operating System Fragments

A set of operating system component prototypes could be both reliable and portable if written in Ada. Such components would include discretionary and mandatory access control at the B3 level, and account for uniprocessors, multi-processors, and multi-computer systems. These prototypes could initially support one local area network. However, their design and implementation might not preclude, nor make more difficult their interaction with other Local Area Networks (LAN's). A prototype set of operating system components could have the following attributes. fault tolerance, survivability, multi-level security at or above the B3 level, portability of applications and system software across a wide variety of machine sizes and types, single and multi-thread machines, multiple priorities, real-time processing, and fully integrated databases.

Operating system prototyping and developmental activities could concern a variety of topics, including:

- Kernel operating systems.
- Bindings to standard systems.
- Program execution support module.
- File manager.
- Authentication server.
- Time synchronization agent.
- Transaction manager.
- Inter-process communication support.
- Alias processes supporting access to remote LAN's.
- Print server.
- Input/output drivers.
- Multi-window system.
- Logging and auditing facilities.

#### A.1.5 Graphics Support

A high-level conceptual model of a graphics support system could be prototyped. The model could be a pipeline showing the flow of information and control between one or more application programs and the graphics display and input hardware. The model could illustrate specific components of a graphics support system. Primarily, these components would be the visual objects (their definition and manipulation), a window management system, and an image generation system.

An object-oriented approach could be adopted in which the application program deals with high-level "visual objects", such as menus and icons, rather than just primitives as points and lines. Possible activities for prototyping specific parts of this model include:

- Visual object specification techniques.
- Interfaces supporting reuse of visual objects.
- Coordination/management support for descriptions employing visual objects.
- Image generation capabilities.
- Window managers.
- Computer graphics interfaces.
- Graphics command, input, output processors.
- Graphics interchange formats.

### **A.1.6 Network Support**

Transfer of a variety of types of information to both local and remote users and systems is a common characteristic of many DoD development environments and end-application systems. Access to these systems' capabilities, both the functions and data, are likely to be accomplished using a LAN architecture in both automated environment and end-application systems. Access to other sites and remote systems and data are likely to be accomplished using intercomputer networking capabilities through the Defense Data Network (DDN).

The needs for network services necessitate tools to build integrated communications subsystems. To realize the benefits of the distributed processing concept, any number of components (hardware and software) from different sources must be able to communicate among themselves by using predetermined protocols. Current DoD protocols are likely to be used in the near term. However, activity in the international standardization community indicates that at some time in the future, ISO protocols may be used for network communication functions. As services are expanded the need for new applications and lower-level protocols grows. Given the experience of the ARPANET, methods which can reduce the complexity and resources needed for new protocol specification and implementation are likely to be necessary for developing and using many DoD systems. Use of the DDN as the long-haul backbone raises the questions of how intercommunications routing will be accomplished when additional factors beyond shortest distance must be accommodated. Prototyping activities in this area could include:

- Ada-based implementation of ISO and DoD transport and internet protocols.
- Generators of Ada protocol software.
- Multi-variable objective functions supporting the optimization of network routing.
- Ada bindings to standards implementations.

## ***A.2 Automated Environment Fragment Capabilities***

Some of the fragments in the repository will only serve the need to develop automated environments. Particularly important in this regard are fragments supporting preliminary and detailed design. Also of interest would be fragments that support software development process definition and project specific customization.

### **A.2.1 Design Description and Analysis**

Careful attention to design is essential to achieving the benefits of software reliability, efficiency, maintainability, and portability. Automated support can substantially improve the design process and the fragments developed in this set of activities will develop currently promising concepts for providing this automated support.

The emphasis in the near term is upon design. This allows early results by capitalizing on technology that has apparent value but has not been brought to the Ada arena. It also provides foundation for later work on tools supporting other life cycle phases.

Example activities in this area are:

- Advanced Ada Program Design Languages (PDL's).
- PDL support tools.
- Graphics support tools for software design.
- Automatic code generation from design.
- Reverse design description from existing code.

## ***A.3 Foundation Deliveries [R]***

By the end of CY 1988, over twenty-five foundation contracts had been completed. These contracts have produced a significant base of fragments that show innovative use of Ada and are im-

mediately useful in Ada software engineering environments. These capabilities were demonstrated at two separate foundation conferences held during 1988.

The Naval Research Laboratory (NRL) in Washington D.C has agreed to manage and coordinate the foundation deliveries. A computer repository has been established to hold the deliveries. Interested parties will be permitted access by submitting a request form to the NRL Research Computer Division. A list of the fragments delivered is maintained on the repository. These foundation deliveries are also available from the IBM and Boeing STARS prime repositories.

## Appendix B. SHADOW PROJECTS [R]

From CY 1985 through CY 1988, the STARS program funded several projects to "shadow" planned DoD application development. These projects have resulted in the development and delivery of successful MCCR software written in Ada. The funded projects were:

- Common Ada Missile Packages (CAMP)

CAMP applied reusable Ada software concepts to DoD missile systems. CAMP also developed and demonstrated a prototype DoD software composition system.

- Ada Based Signal Processing

This project was a software conversion demonstration of a data-flow graphic form technique for building adaptable systems for signal processing.

- Ada Based Integrated Control System (ABICS)

ABICS enables the development and application of Ada software to flight-critical avionics systems. This project demonstrated the application of reusable software parts to the Advanced Tactical Fighter.

- Pilot Ada Capabilities Transition (PACT) Projects

Candidate PACT projects were identified from each service by the Software Engineering Institute (SEI). The projects selected were using commercially available Ada environments and state-of-the-practice software engineering techniques. This initial work helped establish a baseline for measuring future STARS progress and provided lessons learned to help evolve the Ada software engineering process.

- Several service projects were supported under the technology thrust of applying Ada:

- Air Force F-111 Digital Flight Control System
- Air Force Advanced Millimeter Wave Seeker
- Air Force F15 APG8 Radar System
- Army Distributed Computer Design System (DCDS)
- Navy Multi-Ping Computer Program AN/SQS53C
- Navy Command and Control Processor

Presentations on the results of these projects have been made at the STARS workshops and Ada conferences. Specific information on the success of these projects is available from the project managers.

## Appendix C. PRIME CONTRACTORS [R]

### *B.1 Consolidated Plan [R]*

In 1988, three prime STARS contractors were selected: Boeing Aerospace Company, International Business Machines Corporation and the Unisys Corporation. The contractors will operate in associate contractor roles in which the plans, approaches and results will be shared in an effort to find the best solution to the problems being addressed by the STARS program. A plan was prepared early in the contract period that consolidated the three separate plans into a single consolidated five year plan [R].

The thrust of the consolidated plan is to produce a technology base and demonstrate the potential of that base. The success of the plan depends on obtaining the support of the Software Engineering community both within industry and the government. Much of the effort will involve adapting technology which has been or will be developed and integrating it into a common base. This plan does not provide a total solution. The unique environment capabilities necessary to support application domains and fully populated application domain component libraries will not be provided at the present funding level. The focus will be on a common portable base for environment capabilities and common cross domain capabilities and components. The emphasis on adaptability will allow groups outside of STARS to use the products and add to them, extending the environment to provide complete application domain support.

The prime STARS program will produce three major products. a new software life cycle process, an environment, and a repository. The environment will form the base for the STARS products. The new life cycle process definition will be made tangible in the environment through a process manager, a process model and an object model definition which drives the mechanism. The repository will be an instance of the environment instantiated for the reuse process. The environment itself is an adaptable mechanism providing a common, portable basic set of functions and an adaptable set of capabilities necessary to support software engineering.

### *B.2 First Increment Deliveries [R]*

During the first increment of the STARS prime contracts, over two hundred deliveries were made. These deliveries contain a significant body of experiences and development. They provide a basis for the subsequent STARS prime contractor increments. The deliveries are available on the STARS prime contractor repositories for review and comment.